JOURNAL OF COMPUTER AND SYSTEM SCIENCES 34, 87–128 (1987)

# Basic Tree Transducers*

HEIKO VOGLER

*Lehrstuhl fuer Informatik II, RWTH Aachen,*
*Buechel 29-31, D-5100 Aachen, Federal Republic of Germany*

Received January 29, 1986; revised September 29, 1986

The concept of basic tree transducer is investigated; it is obtained as a natural restriction of the concept of macro tree transducer by forbidding nesting of states. Basic tree transducers are characterized in terms of one-turn pushdown machines. A close connection between path languages of ranges of (compositions of) basic tree transducers and (iterated) control on linear grammars is established. This connection allows to prove the strictness of the composition hierarchy of basic tree transducers. © 1987 Academic Press, Inc.

## 1. INTRODUCTION

It is a usual matter of theoretical study to restrict the resources of objects like grammars, automata, and transducers in order to obtain a better understanding of the way they work. Mostly, this study leads to a rich structuring of the well-known main objects and their results. We mention two examples of such investigations.

(a) OI macro grammars are equivalent to nested stack automata [Fis, Aho1/2]. By forbidding nesting of nonterminals in macro grammars, basic macro grammars are defined [Fis, Dow]. In [EngSchvLe] extended basic macro grammars are characterized by stack-pushdown machines. The extension of the grammars allows the parameter positions of nonterminals to keep finite sets of strings rather than only one string. The stack-pushdown machine can be considered as a nested stack automaton, in which the nesting depth of stacks is limited to one.

(b) OI context-free tree grammars [Rou2, EngSch, Dam] are the "tree version" of OI macro grammars, and the path languages of OI context-free tree languages are context-free languages [Rou1, Cou, EngSlu1]. Basic tree grammars [EngSlu1] are context-free tree grammars without nesting of nonterminals. It is shown in [EngSlu1] that the path languages of basic tree languages are linear languages.

The present paper provides a modest approach to study the resources of macro

* The work has been carried out during the author's stay at the Department of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, and it has been supported by the Netherlands Organization for the Advancement of Pure Research (ZWO).



0022-0000/87 $3.00

tree transducers [Eng3, CouFra1, EngVog1] in the same spirit as in the investigations mentioned under (a) and (b).

We investigate nondeterministic and total deterministic basic tree transducers, which are natural restrictions of macro tree transducers in the sense that nesting of states is not allowed. In [CouFra2] total deterministic basic tree transducers are called nonnested recursive schemes. Actually, macro tree transducers are related to basic tree transducers in the same way as context-free tree grammars (and macro grammars) are related to basic tree grammars (and to basic macro grammars, respectively).

Since, conceptually, macro tree transducers combine the features of context-free tree grammars and top-down tree transducers, basic tree transducers can be understood as a combination of basic tree grammars and top-down tree transducers. The involved concept of top-down tree transducer imposes a restriction on the derivation of the tree grammars; the derivations are syntax-directed by an input tree. It is intuitively clear that basic tree transducers are more powerful than top-down tree transducers, because the states may keep a kind of context information. Let us now give an informal, preliminary description of the basic tree transducer model.

A basic tree transducer $M$ consists of three ranked alphabets $N$, $\Sigma$, and $\Delta$ of states, input symbols, and output symbols, respectively, an initial state $A_{\text{in}}$ of rank 0, and a finite set of rules of the form

$$A\langle x\rangle(y_1,\ldots,y_k)\to \textit{if } \operatorname{root}(x)=\sigma \textit{ then } \zeta, \qquad (*)$$

where $A$ is a state of rank $k$ with $k\geqslant 0$, $x$ is the syntactic parameter (of type input tree), $y_1,\ldots,y_k$ are formal parameters (of type output tree), $\sigma$ is an input symbol of some rank $m$ with $m\geqslant 0$, and $\zeta$ is a tree built up from output symbols, the parameters $y_1,\ldots,y_k$, which are viewed as symbols of rank 0, and trees of the form $B\langle \operatorname{sel}_i(x)\rangle(\zeta_1,\ldots,\zeta_n)$ such that $B$ is a state of rank $n\geqslant 0$, $\operatorname{sel}_i(x)$ with $1\leqslant i\leqslant m$ denotes the instruction "select the $i$th subtree of the actual value of $x$," and $\zeta_1,\ldots,\zeta_n$ are trees over output symbols and $y_1,\ldots,y_k$. E.g., $A\langle x\rangle(y_1,y_2)\to \textit{if } \operatorname{root}(x)=\sigma \textit{ then } \delta(\alpha, B\langle \operatorname{sel}_1(x)\rangle(\gamma(\beta), y_2), C\langle \operatorname{sel}_2(x)\rangle)$ is a rule of a basic tree transducer, where $A$, $B$, and $C$ are states of rank 2, 2, and 0, respectively, $\sigma$ is an input symbol of rank 2, and $\delta$, $\gamma$, $\alpha$, $\beta$ are output symbols of rank 3, 1, 0, and 0, respectively. The fact that, in a tree $B\langle \operatorname{sel}_i(x)\rangle(\zeta_1,\ldots,\zeta_n)$, no states may appear in $\zeta_1,\ldots,\zeta_n$, means that nesting of states is not allowed (as it is in macro tree transducers).

The translation induced by a basic tree transducer is defined by means of a tree rewriting relation on trees over output symbols and trees of the form $A\langle s\rangle(t_1,\ldots,t_k)$, where $A$ is a state of rank $k\geqslant 0$, $s$ is in $T_\Sigma$, i.e., a tree over $\Sigma$, and $t_1,\ldots,t_k$ are in $T_\Delta$. In such a tree $\xi$, a rule like $(*)$ can be applied to an occurrence of $A\langle s\rangle(t_1,\ldots,t_k)$, if $s=\sigma(s_1,\ldots,s_m)$ for some trees $s_1,\ldots,s_m$ in $T_\Sigma$. The result of the application is obtained from $\xi$ by replacing $A\langle s\rangle(t_1,\ldots,t_k)$ by $\zeta$, in which every $\operatorname{sel}_i(x)$ $(1\leqslant i\leqslant m)$ and every $y_j$ $(1\leqslant j\leqslant k)$ has been substituted by $s_i$ and $t_j$, respectively. In the usual way, this rewriting induces a relation on $T_\Sigma\times T_\Delta$, which is called

the translation of $M$ and denoted by $\tau(M)$ (recall that the initial state is of rank 0); a pair $(s, t)$ is in $\tau(M)$ if $A_{in}\langle s\rangle$ can be rewritten to $t$. The class of translations induced by basic tree transducers is denoted by BT(TR) (this denotation will be explained in a few moments).

The three main results of this paper are the following, in which the class of recognizable tree languages is denoted by RECOG.

(1) Basic tree transducers are equivalent to one-turn pushdown machines. In the total deterministic case, in which BT(TR) is a class of mappings, compositions of basic tree transducers are equivalent to iterated one-turn pushdown machines.

(2) The images of RECOG under tree-to-path translations of (compositions of) basic tree transducers are generated by (iterated) controlled linear grammars.

(3) The composition hierarchy of (total deterministic) basic tree transducers is strict. This even holds for the classes of images of RECOG.

Before discussing these results in more detail, we briefly recall the concept of "grammar with storage" from [Eng6]. Actually, this tool will be used to define basic tree transducers formally and to describe and prove the results.

In general, a machine consists of a "control" or "program" and a "storage" $S$. By means of predicates and instructions of $S$, the control can test and transform the configurations of $S$. Clearly, if flowcharts are chosen as control, then the usual sequential machines are obtained. In [Eng6] this concept was generalized by allowing context-free grammars as control. In such a "grammar with storage" $M$, every nonterminal has one parameter with configurations of $S$ as actual values. Then, e.g., $aA\langle c\rangle\, bD\langle c'\rangle$ can be a sentential form of $M$, where $a$ and $b$ (and $A$ and $D$) are terminals (and nonterminals, respectively) of the underlying grammar, and $c$ and $c'$ are configurations of the storage. If $A \to aBC$ is a rule of a context-free grammar, where $A$, $B$, and $C$ are nonterminals and $a$ is a terminal, then

$$A\langle x\rangle \to \textit{if } p(x) \textit{ then } aB\langle f(x)\rangle\, C\langle g(x)\rangle \qquad (**)$$

could be a rule of $M$, where $x$ represents the "storage parameter" of $A$, $p$ is a predicate of $S$, and $f$ and $g$ are instructions of $S$. The sentential form $aA\langle c\rangle\, bD\langle c'\rangle$ can be rewritten by rule (**) if $p$ is true for $c$. Then, the resulting sentential form is $aaB\langle c_1\rangle\, C\langle c_2\rangle\, bD\langle c'\rangle$, where $c_1$ and $c_2$ are obtained from $c$ by applying the instructions $f$ and $g$, respectively. Clearly, such a grammar with storage induces a translation from the set of configurations of $S$ to strings over the terminal alphabet of the grammar: for a configuration $c$ of $S$ and a terminal string $w$, $(c, w)$ is in this translation if $A_{in}\langle c\rangle$ can be rewritten to $w$, where $A_{in}$ denotes the initial nonterminal of $M$.

It is obvious that, in the same manner, tree grammars can be used as control. Since a grammar $M$ with storage induces a translation, we also call $M$ an $X(S)$-transducer, where $X$ is the used class of grammars and $S$ is the involved storage. The corresponding class of translations is denoted by $X(S)$. Then macro tree transducers and basic tree transducers are examples of $X(S)$-tranducers: they are con-

text-free tree grammars and basic tree grammars, respectively, with the "tree storage," which is denoted by TR. The configurations of TR are trees over a ranked (input) alphabet. The predicates of TR have the form root $= \sigma$, where $\sigma$ is an input symbol, by means of which the label of the root of a configuration can be tested; an instruction has the form $\text{sel}_i$ which selects the $i$th subtree from a configuration. Note that in our earlier, informal, description of basic tree transducer, the states of the transducer correspond to the nonterminals of the basic tree grammar; in the rule (*), the syntactic parameter $x$ of $A$ is the "storage parameter," whereas the parameters $y_1,\ldots, y_k$ correspond to those of $A$ in the basic tree grammar. Note also that we wrote $\text{root}(x) = \sigma$ rather than the, more obscure, $(\text{root} = \sigma)(x)$. Let CFT and BT denote the classes of context-free tree grammars and of basic tree grammars, respectively. Then, CFT(TR)-transducers are (notational variations of) macro tree transducers and CFT(TR) is the class of translations induced by macro tree transducers. Basic tree transducers are defined as BT(TR)-transducers (cf. Definition 3.3), and this explains the denotation BT(TR) for the class of induced translations.

Now we discuss the results (1), (2), and (3) a bit more in detail.

(1) In [EngVog2] the classes $\text{CFT}_{\text{ext}}(\text{TR})$ and CFT(TR) are characterized by certain regular machines, which are called pushdown machines. The class $\text{CFT}_{\text{ext}}(\text{TR})$ is induced by extended macro tree transducers in which a restricted use of an identity instruction on the configurations of TR is allowed. Expressed in the discussed formalism of "grammars with storage," the pushdown machines are regular tree grammars with the storage P(TR). The use of *regular* tree grammars as control explains the notion of *regular* machine. The configurations of P(TR) are pushdowns in which every square contains besides a usual pushdown symbol also a pointer to a node of the input tree. We refer the reader to Section 3.3 of [EngVog2] for a detailed discussion of P(TR). It is proved in [EngVog2] that $\text{CFT}_{\text{ext}}(\text{TR}) = \text{RT}(\text{P}(\text{TR}))$ and $\text{CFT}(\text{TR}) = \text{RT}(\text{P}_{\text{bex}}(\text{TR}))$, where RT denotes the class of regular tree grammars and $\text{P}_{\text{bex}}$ is a restriction of P called bounded excursion pushdown. We will prove that basic tree transducers can be simulated by *one-turn pushdown machines*, i.e., $\text{RT}(\text{P}_{1t}(\text{TR}))$-transducers, where "1t" indicates that the involved pushdowns are one-turn in the sense of [GinSpa1]. However, the other inclusion does not hold, and thus, basic tree transducers have to be extended slightly such that they fit nicely into the scheme "basic = one-turn." We consider two extensions, viz., regular extension and finite extension. In regular extended basic tree transducers, the parameter positions of nonterminals are allowed to keep a recognizable tree language rather than only one tree. In finite extended basic tree transducers the parameter positions may hold finite tree languages. The corresponding classes of induced tree translations are denoted by $\text{BT}_{\text{reg}}(\text{TR})$ and $\text{BT}_{\text{fin}}(\text{TR})$. Then we prove that $\text{BT}_{\text{reg}}(\text{TR}) = \text{RT}(\text{P}_{1t}(\text{TR}))$ and $\text{BT}_{\text{fin}}(\text{TR}) = \text{RT}(\text{P}_{1t,\text{bex}}(\text{TR}))$ (cf. Theorem 4.12(i)). Hence, the "non-nesting" restriction on macro tree transducers corresponds to the "one-turn" restriction on pushdown machines.

A machine characterization of BT($S$) itself turns out to be more involved.

However, in the total deterministic case, in which the translations of basic tree transducers are mappings of type $T_\Sigma \to T_\Delta$, the extensions do not increase the power of basic tree transducers. Thus, we obtain the machine characterization $D_t BT(TR) = D_t RT(P_{1t}(TR)) = D_t RT(P_{1t,bex}(TR))$, where $D_t$ indicates that the transducers are total deterministic. In the total deterministic case even the composition of basic tree transducers can be characterized: by iterated one-turn pushdown machines (cf. Theorem 4.12(ii)). Such regular machines have an iterated one-turn pushdown as storage, of which a configuration is a pushdown of pushdowns of ... of pushdowns of pointers to nodes of trees and every involved pushdown is one-turn. For the concept of iterated pushdown (without the one-turn restriction) we refer to [Gre2, Mas, DamGoe, Eng5, EngVog3].

The characterization of extended basic tree transducers by one-turn pushdown machines is closely related to the equivalence of extended basic macro grammars and stack-pushdown machines as mentioned under (i). Roughly speaking, extended basic tree transducers and one-turn pushdown machines are the tree versions of extended basic macro grammars and stack-pushdown machines, respectively, with syntax-directed derivations and computations, respectively.

(2) Let $\pi$ denote the class of those relations $\pi_\Delta$, that associate with every tree over $\Delta$ the set of its paths [Roul, Cou, EngSlu1]. In [Eng3] is was claimed that $\pi(CFT(TR)(RECOG))$ is contained in the class of images of RECOG under top-down tree-to-string transducers. (Note that $\pi(CFT(TR)(RECOG))$ denotes the class of path languages of images of RECOG under CFT(TR)-transducers.) In the formalism of "grammars with storage," top-down tree-to-string transducers are described as CF(TR)-transducers (CF denotes the class of context-free grammars). Hence, the claim of [Eng3] is: $\pi(CFT(TR)(RECOG)) \subseteq CF(TR)(RECOG)$. We give a formal proof of this claim and also show that $\pi(BT(TR)(RECOG)) \subseteq LIN(TR)(RECOG)$ (LIN denotes the class of linear grammars). The latter result is obtained by studying the tree-to-path translations of basic tree transducers, i.e., the class $BT(TR) \circ \pi$. In particular, it is observed that the paths of basic tree transducer rules have the form of right-hand sides of rules of linear grammars. We note that the notion of look-ahead on storage types [Eng6, EngVog2/3/4] plays an essential role in the proof of this result. Thus, for paths, the restriction of macro tree transducers to basic tree transducers corresponds to the restriction of CF(TR)-transducers to LIN(TR)-transducers. Note that a similar correspondence also holds for path languages of CFT and BT, see (ii).

In every derivation of a LIN(TR)-transducer only one path of the given input tree is considered. Since $\pi(RECOG)$ is contained in the class $\mathscr{L}_{REG}$ of regular languages, it is therefore intuitively clear that LIN(TR)(RECOG) can be generated by $\mathscr{L}_{REG}$-controlled linear grammars [GinSpa2, Gre3/4, Kha1/2, DusPar, Vog1], i.e., by linear grammars. Denote the class of languages generated by $\mathscr{L}$-controlled linear grammars by $CTRL(LIN, \mathscr{L})$. We prove that $\pi(BT(TR)(RECOG)) \subseteq CTRL(LIN, \mathscr{L}_{REG})$ and even that, for every $n \geqslant 0$, $\pi(BT(TR)^n(RECOG)) \subseteq CTRL_n(LIN, \mathscr{L}_{REG})$, where $CTRL_0(LIN, \mathscr{L}) = \mathscr{L}$ and for every $k \geqslant 0$,

$CTRL_{k+1}(LIN, \mathscr{L}) = CTRL(LIN, CTRL_k(LIN, \mathscr{L}))$ (cf. Theorem 5.9). Note that $BT(TR)^n$ denotes the class of translations induced by the composition of $n$ basic tree transducers.

(3) The study of tree-to-path translations of basic tree transducers leads quite straightforwardly to the proof of the desired result. In [Gre3] it is shown that the class of controlled linear grammars (viewed as "grammar directed translators" [Gre3]) form a hierarchical operator on classes of languages; in particular, for increasing $n$, the classes $CTRL_n(LIN, \mathscr{L}_{REG})$ form a proper hierarchy. Since $\pi(BT(TR)^n(RECOG)) \subseteq CTRL_n(LIN, \mathscr{L}_{REG})$, we only have to show the existence of a language which is a path language of an element of $D_t BT(TR)^{n+1}(RECOG)$ and which is not contained in $CTRL_n(LIN, \mathscr{L}_{REG})$. Essentially, this sample language is the same as the language defined in [Kha1] to prove that $CTRL_n(LIN, \mathscr{L}_{REG})$ is properly contained in $CTRL_{n+1}(LIN, \mathscr{L}_{REG})$. Thus, we obtain strictness of the hierarchies $\{BT(TR)^n(RECOG) \mid n \geq 0\}$ and $\{D_t BT(TR)^n(RECOG) \mid n \geq 0\}$ (cf. Theorem 5.12).

This paper is divided into 6 sections of which the second recalls all important notions discussed above. In the third section, the model of a basic tree transducer is formally defined and some elementary properties of the class BT(TR) of induced translations are proved. The machine characterizations of the two extensions of BT(TR) and of the composition of total deterministic basic tree transducers are provided in Section 4. Section 5 deals with tree-to-path translations of basic tree transducers, the composition hierarchy, and the relations between top-down tree transducers, basic tree transducers, and macro tree transducers. Finally, Section 6 contains some conclusions and two open problems.

Because of the frequent citation, the reference to the paper [EngVog2] will be abbreviated by [EV].

## 2. Preliminaries

We try to make this paper self-contained. For this purpose, we recall the basic notions concerning relations and substitution (Sect. 2.1), trees, paths of trees, and tree relabeling (Sect. 2.2), grammars (Sect. 2.3), grammars with storage (Sect. 2.4), and controlled linear grammars (Sect. 2.5). However, we assume the reader is familiar with the basics of formal language theory as, e.g., in [HopUll, Har, Sal]. The empty string is denoted by $\lambda$.

### 2.1. *Relations and Substitution*

Let Nat be the set of non-negative integers, i.e., $Nat = \{0, 1, 2, ...\}$. For $n \geq 0$, $[n]$ denotes the subset $\{1, ..., n\}$ of Nat; $[0] = \varnothing$, where $\varnothing$ denotes the empty set. For a set $U$, the set of subsets of $U$ is denoted by $P(U)$.

Let $U$ and $V$ be two arbitrary sets. A function $\tau: U \to V$ is also denoted, as usual, in lambda notation, i.e., $\tau = \lambda u \in U.\tau(u)$. The *difference* of $U$ and $V$, denoted by

$U - V$, is the set $\{u \mid u \in U \text{ and } u \notin V\}$. A *relation* $R$ *from* $U$ *to* $V$ is any subset of $U \times V$. The *domain* and the *range* of $R$, denoted by $\mathrm{dom}(R)$ and $\mathrm{range}(R)$, respectively, are the sets $\{u \mid \text{there is a } v \in V \text{ such that } (u, v) \in R\}$ and $\{v \mid \text{there is a } u \in U \text{ such that } (u, v) \in R\}$, respectively. For a set $A$, the *image of* $A$ *under* $R$, denoted by $R(A)$, is the set $R(A) = \{v \mid \text{there is a } u \in A \text{ such that } (u, v) \in R\}$. For two relations $R_1$ and $R_2$ the *composition* of $R_1$ and $R_2$ is the set $\{(u, w) \mid (u, v) \in R_1 \text{ and } (v, w) \in R_2 \text{ for some } v\}$ which is denoted by $R_1 \circ R_2$. For $n \geqslant 1$, the *n-fold composition* of $R$, denoted by $R^n$, is the relation $R \circ \cdots \circ R$ ($n$ times) and $R^0$ is the identity. The *transitive* and the *reflexive, transitive closure* of $R$ are denoted by $R^+$ and by $R^*$, respectively. The above-mentioned concepts can be extended in an obvious way to classes of relations, for which we use the same notations.

Very often we use an abbreviation for the *substitution* of strings into strings. Let $v$ be a string, let $U$ and $V$ be arbitrary sets, and let $\theta$ be a mapping from $U$ into $P(V)$. Then $v[u \leftarrow \theta(u);\ u \in U]$ denotes the set of strings obtained from $v$ by replacing every occurrence of $u \in U$ in $v$ by an element of $\theta(u)$, where different occurrences of $u$ may be replaced by different elements of $\theta(u)$. If $\theta(u)$ is a singleton, then we do not specify $\theta(u)$, but only provide its element. If $U$ is understood from the context, then the part behind the semicolon is dropped. We only use the notation $v[u \leftarrow \theta(u);\ u \in U]$ if there are no overlapping occurrences of elements of $U$ in $v$.

## 2.2. *Trees, Paths of Trees, and Tree Relabeling*

A *ranked set* $\Delta$ is a (possibly infinite) set in which to every symbol a nonnegative integer is associated that is called its rank. For every $n \in \mathrm{Nat}$, $\Delta_n$ denotes the set of symbols of $\Delta$ which have rank $n$. The rank of a symbol is sometimes indicated as a superscript, e.g., $\delta^{(2)}$ means that $\delta$ is of rank 2. If $\Delta$ is finite, then we obtain the usual concept of a *ranked alphabet*.

Throughout this paper, $\Omega$ denotes a countably infinite ranked set such that every subset $\Omega_n$ is infinite.

Let $\Delta$ be a ranked set and let $V$ and $U$ be arbitrary sets. Then $\Delta\langle V\rangle$ is the ranked set $\{\delta\langle v\rangle^{(n)} \mid \delta \in \Delta_n \text{ with } n \geqslant 0 \text{ and } v \in V\}$. The set of (labeled) *trees over* $\Delta$ *indexed by* $U$, denoted by $T_\Delta(U)$, is the smallest subset of $(\Delta \cup U \cup \mathrm{PC})^*$, where PC denotes the set of left parenthesis, right parenthesis, and comma, such that (i) $U \subseteq T_\Delta(U)$ and (ii) if $\delta \in \Delta_k$ with $k \geqslant 0$ and $t_1, \ldots, t_k \in T_\Delta(U)$, then $\delta(t_1, \ldots, t_k) \in T_\Delta(U)$ (if $n = 0$, then we write $\delta$ rather than $\delta()$). $T_\Delta(\varnothing)$ is the usual set of trees over $\Delta$, also denoted by $T_\Delta$. A tree $t \in T_\Delta(U)$ is *linear* in $U$ if every element of $U$ occurs at most once in $t$. The *height* of a tree $t \in T_\Delta$, denoted by $\mathrm{height}(t)$, is provided by the function height: $T_\Delta \to \mathrm{Nat}$, which is defined inductively on the structure of the tree: (i) For $\delta \in \Delta_0$, $\mathrm{height}(\delta) = 1$. (ii) For $t = \delta(t_1, \ldots, t_n)$ for some $n \geqslant 1$, $\delta \in \Delta_n$, and $t_1, \ldots, t_n \in T_\Delta$, $\mathrm{height}(t) = 1 + \max\{\mathrm{height}(t_i) \mid i \in [n]\}$. The *yield* of a tree $t$, denoted by $\mathrm{yield}(t)$, is the concatenation of its leaf labels. The yield of a relation $R \subseteq U \times T_\Delta$, where $U$ is an arbitrary set and $\Delta$ is a ranked set, is the relation $\mathrm{yield}(R) = \{(u, w) \mid (u, t) \in R \text{ and } \mathrm{yield}(t) = w \text{ for some } t\}$, which is also abbreviated by $yR$. This notion can be extended to classes of relations in an obvious way. Any subset of $T_\Delta$ is a *tree*

*language* and the class of *recognizable (or regular) tree languages* is denoted by RECOG.

Note that a tree over $\Delta$ can be viewed as a string over $\Delta \cup PC$, where PC is as above. In this sense, the abbreviation for substitution, which is defined in Section 2.1, also applies to trees.

We recall the concept of paths of trees (see, e.g., [Roul, Cou, EngSlu1]). Let $\Delta$ be a ranked alphabet. The *path alphabet* associated with $\Delta$, denoted $p(\Delta)$, is the finite set $\{(\delta, 0) \mid \delta \in \Delta_0\} \cup \{(\delta, i) \mid \delta \in \Delta_n \text{ for some } n \geqslant 1 \text{ and } 1 \leqslant i \leqslant n\}$. For every ranked alphabet $\Delta$, the *path translation of* $\Delta$, denoted $\pi_\Delta$, is the smallest subset of $T_\Delta \times p(\Delta)^*$ for which (i) and (ii) hold: (i) For $\delta \in \Delta_0$, $(\delta, (\delta, 0)) \in \pi_\Delta$. (ii) For $t = \delta(t_1, ..., t_k)$ with $\delta \in \Delta_k$, $k \geqslant 1$, and $t_1, ..., t_k \in T_\Delta$, if $(t_i, p) \in \pi_\Delta$ for some $i \in [k]$, then $(t, (\delta, i)p) \in \pi_\Delta$. Let $\pi$ denote the class of path translations $\pi_\Delta$ for some finite subset $\Delta$ of $\Omega$.

2.1. FACT (cf. Corollary 4.12 of [Cou]). *$\pi$(RECOG) is contained in the class of regular languages.*

A *total deterministic bottom-up finite state relabeling* (for short: relabeling) [Eng1] is a tuple $(Q, \Sigma, \Delta, R)$, where $Q$, $\Sigma$, and $\Delta$ are the ranked alphabets of states (of rank 1), input symbols, and output symbols, respectively, and $R$ is the finite set of rules of the form $\sigma(q_1(x_1), ..., q_k(x_k)) \to q(\delta(x_1, ..., x_k))$ with $k \geqslant 0$, $\sigma \in \Sigma_k$, $\delta \in \Delta_k$, $q, q_1, ..., q_k \in Q$, and $x_1, ..., x_k$ are variables of rank 0. Moreover, for every $\sigma \in \Sigma_k$ with $k \geqslant 0$ and $q_1, ..., q_k \in Q$, there is exactly one rule in $R$ with left-hand side $\sigma(q_1(x_1), ..., q_k(x_k))$. These rules turn into tree rewriting rules, as usual, by substituting trees in $T_\Delta$ for the $x$'s. The translation induced by $M$, denoted by $\tau(M)$, is $\{(s, t) \in T_\Sigma \times T_\Delta \mid s \Rightarrow_M^* q(t) \text{ for some } q \in Q\}$, where $\Rightarrow_M$ denotes the derivation relation of $M$; $\tau(M)$ is also called a *relabeling* and the class of relabelings is denoted by RELAB. Note that every relabeling is a mapping.

2.2. FACT. *RECOG is closed under RELAB.*

*Proof.* Relabelings are defined as bottom-up tree transducers, which are linear with respect to the input tree. By Corollary 3.11 of [Eng1], RECOG is closed under linear bottom-up tree transducers. ∎

2.3. *Grammars*

All the grammars, that are used in this paper, are context-free tree grammars or context-free grammars. Here we formally recall from [EngSch] the concept of context-free tree grammar. In order to facilitate the definition, we fix for the rest of this paper the infinite ranked set $Y = \{y_1, y_2, ...\}$ of parameters, in which every element has rank 0; for $n \geqslant 0$, $Y_n = \{y_1, ..., y_n\}$.

A *context-free tree grammar* $G$ is a tuple $(N, \Delta, A_{in}, R)$, where $N$ is the ranked alphabet of nonterminals, $\Delta$ is the ranked alphabet of terminals, $A_{in} \in T_N$ is the initial term, and $R$ is the finite set of rules of the form $A(y_1, ..., y_k) \to \zeta$, where $A \in N_k$, $k \geqslant 0$, and $\zeta \in T_{N \cup \Delta}(Y_k)$. (Note that we allow $G$ to have an initial term

rather than only an initial nonterminal. However, this does not increase the generating power of context-free tree grammars.) A *regular tree grammar* is a context-free tree grammar in which every nonterminal has rank 0. Clearly, the initial term of a regular tree grammar consists, as in the usual definition, just of one nonterminal.

The outside-in (OI-) derivation relation of $G$, denoted by $\Rightarrow_G$, is a binary relation on $T_{N\cup\Delta}$, and it is defined as follows:

For $\xi_1, \xi_2 \in T_{N\cup\Delta}$, $\xi_1 \Rightarrow_G \xi_2$ iff
there is a rule $A(y_1,..., y_k) \to \zeta$ in $R$ for some $k \geqslant 0$, $A \in N_k$, and $\zeta \in T_{N\cup\Delta}(Y_k)$,
there is a $\xi \in T_{N\cup\Delta}(\{z\})$, where $z$ is a symbol of rank 0 which occurs exactly once in $\xi$, and $z$ does not occur in a subtree of the form $B(\zeta_1,..., \zeta_n)$ of $\xi$ (with $B \in N_n$),
and there are $\xi'_1,..., \xi'_k \in T_{N\cup\Delta}$
such that $\xi_1 = \xi[z \leftarrow A(\xi'_1,..., \xi'_k)]$
and $\xi_2 = \xi[z \leftarrow \zeta']$, where $\zeta' = \zeta[y_i \leftarrow \xi'_i;\ i \in [k]]$.
The language generated by $G$, denoted $L(G)$, is the set $\{t \in T_\Delta \mid A_{\text{in}} \Rightarrow_G^* t\}$.

We will also consider infinite context-free tree grammars that may have infinite sets of nonterminals and rules. The notion of OI-derivation relation is defined in exactly the same way as for ordinary, finite grammars.

A *context-free-grammar* $G$ is a tuple $(N, \Sigma, A_{\text{in}}, R)$, where $N$ and $\Sigma$ are the alphabets of nonterminals and terminals, respectively, $A_{\text{in}} \in N$ is the initial nonterminal, and $R$ is the finite set of rules of the form $A \to \zeta$, where $A \in N$ and $\zeta \in (N \cup \Sigma)^*$. $G$ is a *linear grammar (regular grammar)*, if for every rule $A \to \zeta$, $\zeta \in \Sigma^* \cup \Sigma^* N \Sigma^*$ ($\zeta \in \Sigma^* \cup \Sigma^* N$, respectively). In the usual way the derivation relation $\Rightarrow_G$ of a context-free grammar $G$ is defined as a binary relation on $(N \cup \Sigma)^*$. The language generated by $G$ is denoted by $L(G)$.

In the rest of this paper we abbreviate the classes of context-free tree grammars, regular tree grammars, context-free grammars, linear grammars, and regular grammars by CFT, RT, CF, LIN, and REG, respectively. The classes of languages generated by grammars in $X$, where $X$ is one of the discussed classes, is denoted by $\mathscr{L}_X$. Note that $\text{yield}(\mathscr{L}_{\text{RT}}) = \mathscr{L}_{\text{CF}}$, where we assume that there is a specific symbol of rank 0 that is mapped under yield to the empty string. Note also that $\mathscr{L}_{\text{RT}} = \text{RECOG}$.

### 2.4. *Grammars with Storage*

The technical framework for the investigations of this paper is the concept of grammar with storage, introduced in [Eng6]. We briefly recall this concept.

A *storage type* $S$ is a tuple $(C, P, F, I, E, m)$, where $C$ is the set of configurations, $P$, $F$, and $E$ are the set of predicate symbols, instruction symbols, and encoding symbols, respectively, $I$ is the set of input elements, and $m$ is the meaning function, which associates with every $p \in P$ a mapping $m(p): C \to \{\text{true}, \text{false}\}$, with every $f \in F$ a function $m(f): C \to C$, and with every $e \in E$ a function $m(e): I \to C$.

In the rest of this paper $S$ denotes the storage type $(C, P, F, I, E, m)$ if not

specified otherwise. In the usual way $m$ is extended to the set $\mathrm{BE}(P)$ of boolean expressions over $P$, where *true* and *false* are the boolean constants. We use "predicate," "instruction," and "encoding" as shorthands for "predicate symbol," "instruction symbol," and "encoding symbol," respectively,

The *trivial storage type* $S_0$ is the tuple $(\{c\}, \varnothing, \{\mathrm{id}\}, \{c\}, \{\mathrm{id}\}, m)$, where $c$ is an arbitrary object and $m(\mathrm{id})(c) = c$.

$S_{\mathrm{id}}$ denotes the storage type $(C, P, F \cup \{\mathrm{id}\}, I, E, m')$, where id is a new instruction, $m'$ restricted to $P \cup F \cup E$ is $m$, and for every $c \in C$, $m'(\mathrm{id})(c) = c$.

In the sequel let $X$ range over $\{\mathrm{CFT}, \mathrm{RT}, \mathrm{CF}, \mathrm{LIN}, \mathrm{REG}\}$. A grammar in $X$ is also called $X$-grammar. After having defined the concept of basic tree grammar (cf. Definition 3.1), we will use the following definitions and concepts also for $X = \mathrm{BT}$, where BT denotes the class of basic tree grammars.

A grammar with storage induces a translation from the input set of the storage type to the set of terminal trees or strings of the involved grammar. Hence, we call such a device an $X(S)$-transducer, where $X$ is the class of used grammars and $S$ is the storage type. Recall that an $X$-grammar is specified by a tuple $(N, \Delta, A_{\mathrm{in}}, R)$; for $X = \mathrm{CFT}$, $A_{\mathrm{in}}$ is an initial term, otherwise $A_{\mathrm{in}}$ is an initial nonterminal. The "storage parameter" $x$ (cf. discussion in the Introduction) will be dropped from the rules of an $X(S)$-transducer, because it is clear where it should occur.

An *$X(S)$-transducer* $M$ is a tuple $(N, e, \Delta, A_{\mathrm{in}}, R)$, where $N$, $\Delta$, and $A_{\mathrm{in}}$ are the alphabets of nonterminals, terminals, and the initial term (or the initial nonterminal) as defined for $X$-grammars, $e \in E$ is the encoding of $M$, and $R$ is the finite set of rules; each rule has the form $\Theta \to \textit{if } b \textit{ then } \zeta$, where $\Theta \to \xi$ is a rule of a usual $X$-grammar, $b \in \mathrm{BE}(P)$, and $\zeta$ is obtained from $\xi$ by replacing every occurrence of a nonterminal $B$ by $B\langle f \rangle$ for some $f \in F$, i.e., $\zeta \in \xi[B \leftarrow B\langle F \rangle; B \in N]$. We also allow sequences $f_1; \ldots; f_\kappa$ $(\kappa \geq 1)$ of instructions to be associated with nonterminals in the right-hand side of a rule. (By introducing auxiliary nonterminals such sequences can easily be reduced to length one.) $M$ is *deterministic* if for every $c \in C$ and every two different rules $\Theta \to \textit{if } b_1 \textit{ then } \zeta_1$ and $\Theta \to \textit{if } b_2 \textit{ then } \zeta_2$, $m(b_1)(c) = \mathrm{false}$ or $m(b_2)(c) = \mathrm{false}$.

If $r$: $\Theta \to \textit{if } b \textit{ then } \zeta$ is a rule of an $X(S)$-transducer $M$ and $A$ is the nonterminal in $\Theta$, then $r$ is called $(A, b)$-rule or $A$-rule of $M$, $b$ is the test of $r$, and $\zeta$ is the right-hand side of $r$. If $b = \textit{true}$, then we abbreviate $r$ by $\Theta \to \zeta$. For a finite set $U$, which is ranked in case $X \in \{\mathrm{CFT}, RT\}$, $M(U)$ denotes the $X(S)$-transducer $(N, e, \Delta \cup U, A_{\mathrm{in}}, R)$.

The *$X$-grammar* $G(M) = (N', \Delta, A, R')$ *associated with* $M$ is defined by $N' = N\langle C \rangle$, $A$ is any element of $N'$, and $R'$ is obtained as follows. If $\Theta \to \textit{if } b \textit{ then } \zeta$ is in $R$, then for every $c \in C$ such that $m(b)(c) = \mathrm{true}$ and such that, for every instruction $f$ occurring in $\zeta$, $m(f)$ is defined on $c$, the rule $\Theta_c \to \zeta_c$ is in $R'$, where $\Theta_c = \Theta[A \leftarrow A\langle c \rangle; A \in N]$ and $\zeta_c = \zeta[B\langle f \rangle \leftarrow B\langle m(f)(c) \rangle; B \in N, f \in F]$. Note that an associated grammar may have infinitely many nonterminals and infinitely many rules.

The derivation relation of $M$, denoted $\Rightarrow_M$, is defined by $\Rightarrow_M = \Rightarrow_{G(M)}$. The *translation induced by* $M$, denoted by $\tau(M)$, is the set $\{(u, v) \mid u \in I, v \in \Phi, \text{ and}$

$A_{\text{in}}\langle m(e)(u)\rangle \Rightarrow_M^* v\}$, where $\Phi$ is, according to the type of $G(M)$, either $\Delta^*$ or $T_\Delta$, and for every $c \in C$, $A_{\text{in}}\langle c\rangle = A_{\text{in}}[A \leftarrow A\langle c\rangle;\ A \in N]$. $M$ is *total*, if $\text{dom}(\tau(M)) = \text{dom}(m(e))$. Two $X(S)$-transducers are *equivalent* if they induce the same translation. The class of translations induced by (total deterministic) $X(S)$-transducers is denoted by $X(S)$, ($D_t X(S)$, respectively).

By considering the trivial storage type $S_0$, an $X(S_0)$-transducer can be regarded again as an $X$-grammar. Clearly, $\text{range}(X(S_0)) = \mathscr{L}_X$.

Note that $\text{yRT}(S) = \text{CF}(S)$. Note that $\text{REG}(S)$-transducers are very close to usual (one-way) $S$ automata, which have a finite control, a one-way input tape, and an auxiliary storage of type $S$ (cf. [Eng6]). Actually, $\text{range}(\text{REG}(S)) = \mathscr{L}(S)$, where $\mathscr{L}(S)$ denotes the class of languages accepted by one-way $S$ automata. Furthermore, $\text{RT}(S)$-transducers can be considered as top-down tree automata with an additional storage of type $S$. Such automata accept $\text{range}(\text{RT}(S))$. In this sense, e.g., the usual finite top-down tree automata (see, e.g., [ThaWri, Eng1]) coincide with $\text{RT}(S_0)$-transducers and the pushdown tree automata of [Gue] coincide with $\text{RT}(\text{P})$-transducers, where P denotes the pushdown storage type.

An $X(S)$-transducer $M$ is in *standard test form* if for every two different tests $b_1$ and $b_2$, which occur in $M$, $b_1$ and $b_2$ are mutually exclusive, i.e., for every $c \in C$, $m(b_1 \text{ and } b_2)(c) = \text{false}$; in other words, if there is a $c \in C$ such that two tests $b_1$ and $b_2$ are true on $c$, then $b_1 = b_2$. The tests of $M$ are called *standard tests*. (Note that this notion of standard form is slightly more general than the one in Definition 3.10 of [EV].)

2.3. FACT (cf. Lemma 3.11 of [EV]). *Every $X(S)$-transducer can be equivalently transformed into standard test form. Determinism and totality are preserved.*

2.5. *Controlled Linear Grammars*

We recall the concept of controlled linear grammar [GinSpa2, Gre3/4, Kha1/2, ParDus, Vog1].

A *labeled linear grammar* $G$ is a tuple $(N, \Delta, A_{\text{in}}, R, \phi, \Sigma)$, where $G' = (N, \Delta, A_{\text{in}}, R)$ is a linear grammar (called the underlying grammar), $\Sigma$ is a finite set of labels, and $\phi$ is a mapping from $\Sigma$ onto $R$ (i.e., for every $r \in R$, there is a $\sigma \in \Sigma$ such that $\phi(\sigma) = r$). A labeled rule is denoted by $\sigma: A \to \zeta$ if $\phi(\sigma) = (A \to \zeta)$. Note that a rule may have different labels. For $\xi_1, \xi_2 \in \Delta^* N \Delta^* \cup \Delta^*$, we write $\xi_1 \Rightarrow_{G',\sigma} \xi_2$, if $\xi_1 \Rightarrow_{G'} \xi_2$ by the use of the rule with label $\sigma$. The notation is extended in an obvious way to $\Rightarrow_{G',u}$ for $u \in \Sigma^*$.

A *controlled linear grammar* $K$ is a tuple $(G, H)$, where $G$ is a labeled linear grammar with label set $\Sigma$, and $H$ is a language over $\Sigma$, i.e., $H \subseteq \Sigma^*$. The *language generated* by $K$, denoted by $L(K)$, is the set $\{w \in \Delta^* \mid A_{\text{in}} \Rightarrow_{G',u} w \text{ for some } u \in H\}$, where $G'$ is the underlying grammar with terminal alphabet $\Delta$. Since $G$ and $G'$ are so closely related, we will use $\Rightarrow_{G,u}$ rather than $\Rightarrow_{G',u}$. $H$ is the *control language* of $K$ and every $u \in H$ is a *control string*.

Let $\mathscr{L}$ be a class of (string) languages. A controlled linear grammar is an *$\mathscr{L}$-controlled linear grammar* if the control language is a member of $\mathscr{L}$. The class of

languages generated by $\mathscr{L}$-controlled linear grammars is denoted by CTRL(LIN, $\mathscr{L}$). As a special case of $\mathscr{L}$-controlled linear grammar, an *$\mathscr{L}$-controlled regular grammar* is defined in the obvious way; then the underlying grammar is regular. The class of languages generated by $\mathscr{L}$-controlled regular grammars is denoted by CTRL(REG, $\mathscr{L}$). Note that $\mathscr{L}$-controlled regular grammars are very close to generalized sequential machines (for short: gsm) and CTRL(REG, $\mathscr{L}$) = GSM($\mathscr{L}$), where GSM is the class of gsm-mappings (cf. e.g., [HopUll] for a formal definition of gsm).

The mechanism of controlling a linear grammar can be iterated as follows: given a class $\mathscr{L}$ of languages, then $\mathrm{CTRL}_0(\mathrm{LIN}, \mathscr{L}) = \mathscr{L}$ and for every $n \geqslant 0$, $\mathrm{CTRL}_{n+1}(\mathrm{LIN}, \mathscr{L}) = \mathrm{CTRL}(\mathrm{LIN}, \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}))$. In [Kha1] it is shown that $\{\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{CF}}) \mid n \geqslant 0\}$ is an infinite strict hierarchy. Since CTRL(LIN, –) is a hierarchical operator on every full semi-AFL $\mathscr{L}$ for which $\mathscr{L} \neq \mathrm{CTRL}(\mathrm{LIN}, \mathscr{L})$ (cf. Corollary 6.6 of [Gre2]), also $\{\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}}) \mid n \geqslant 0\}$ is an infinite strict hierarchy. Note that a full semi-AFL is a family of languages closed under intersection with regular sets, inverse homomorphisms, homomorphisms, and union (cf., e.g., [Gin]).

Since linear grammars are left derivation bounded, we obtain, as an instance of Theorem 3.5 of [Gre2], an AFL-theoretic property of $\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L})$.

2.4. FACT. *If $\mathscr{L}$ is a full semi-AFL, then for every $n \geqslant 0$, $\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L})$ is a full semi-AFL.*

The classes CTRL(LIN, $\mathscr{L}$) and CTRL(REG, $\mathscr{L}$) can also be expressed in terms of images of $\mathscr{L}$ under LIN(one-way)-transducers and REG(one-way)-transducers, respectively, where one-way [Eng6, Vog2] is the storagetype $(C, P, F, I, E, m)$ and $C = \Xi^*$, $\Xi$ is a fixed infinite set of symbols, $P = \{\mathrm{sym} = a \mid a \in \Xi\} \cup \{\mathrm{empty}\}$, $F = \{\mathrm{read}\}$, $I = \Xi^*$, $E = \{\Sigma \mid \Sigma \text{ is a finite subset of } \Xi\}$, and for every $bw \in \Xi^*$ with $b \in \Xi$, $m(\mathrm{sym} = a)(bw) = \mathrm{true}$ iff $(b = a)$ and $m(\mathrm{sym} = a)(\lambda) = \mathrm{false}$, $m(\mathrm{empty})(bw) = \mathrm{false}$ and $m(\mathrm{empty})(\lambda) = \mathrm{true}$, $m(\mathrm{read})(bw) = w$ and $m(\mathrm{read})(\lambda)$ is undefined, $m(\Sigma) = \lambda w \in \Sigma^*.w$. Note that the storage-type one-way does not have an identity.

The two differences between controlled linear grammars and LIN(one-way)-transducers are the following. First, the former device has to consume the whole control string, whereas the latter device can ignore suffixes of the input, and second, the former device cannot detect the end of the control string whereas the latter can. The first difference can be overcome by forcing every LIN(one-way)-transducer to read its whole input string (using an additional nonterminal). The second difference is overbridged by requiring that the class $\mathscr{L}$ of languages, of which the images are studied, is closed under right-marking and finite substitution. E.g., these closure properties are guaranteed if $\mathscr{L}$ is a full semi-AFL. The (inductive) proof of the next lemma is left to the reader.

2.5. LEMMA. *For every $n \geqslant 0$ and every full semi-AFL $\mathscr{L}$, $\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}) = \mathrm{LIN(one\text{-}way)}^n(\mathscr{L})$ and $\mathrm{CTRL}_n(\mathrm{REG}, \mathscr{L}) = \mathrm{REG(one\text{-}way)}^n(\mathscr{L}) = \mathscr{L}$.*

In Proposition 2.2 of [Gre2] it is stated that regular control languages do not influence the power of the controlled class of grammars.

2.6. FACT. LIN(one-way)($\mathscr{L}_{\mathrm{REG}}$) = $\mathscr{L}_{\mathrm{LIN}}$.

## 3. BASIC MODEL

In this section we formally introduce basic tree transducers and prove some elementary properties. As discussed in the Introduction, a basic tree transducer can be considered as a basic tree grammar of which the derivations are syntax-directed by an input tree. Hence, we formalize basic tree transducers as BT(TR)-transducers, where BT is the class of basic tree grammars [EngSlu1] and TR is the tree storage type [Eng6, EV].

A basic tree grammar is a context-free tree grammar in which the initial term consists just of one nonterminal and in which the nonterminals may not occur nested in the right-hand side of rules.

3.1. DEFINITION. A *basic tree grammar* $G$ is a context-free tree grammar $(N, \Delta, A_{\mathrm{in}}, R)$ such that $A_{\mathrm{in}} \in N_0$ and for every rule $A(y_1,..., y_k) \to \zeta$ with $k \geqslant 0$ in $R$, $\zeta \in T_\Delta(Y_k \cup \Psi_k)$, where $\Psi_k = \{B(t_1,..., t_n) \mid B \in N_n,\ n \geqslant 0,$ and $t_1,..., t_n \in T_\Delta(Y_k)\}$.

Note that every path through a right-hand side of a rule contains at most one nonterminal.

The class of basic tree grammars is denoted by BT. Clearly, RT $\subseteq$ BT $\subseteq$ CFT. Since basic tree grammars are special context-free tree grammars, their derivation relation is defined in outside-in mode. However, every OI-derivation of a basic tree grammar is also an inside-out derivation (in the usual sense of [Fis, EngSch]), because the parameters of the rewritten nonterminals always contain terminal trees.

The tree storage type has trees as configurations. By means of predicates of the form "root $= \sigma$" the label of the root of a tree can be tested and the instruction "sel$_i$" selects the $i$th subtree of a tree. We recall the formal definition from [Eng6].

3.2. DEFINITION. The *tree storage type* TR is the storage type $(C, P, F, I, E, m)$, where $C = T_\Omega$, where $\Omega$ is the infinite ranked set mentioned in Section 2.2, $P = \{\mathrm{root} = \sigma \mid \sigma \in \Omega\}$, $F = \{\mathrm{sel}_i \mid i \geqslant 1\}$, $I = C$, $E = \{\Sigma \mid \Sigma$ is a finite subset of $\Omega\}$, and for every $t = \delta(t_1,..., t_k)$ in $C$ with $\delta \in \Omega$ of rank $k \geqslant 0$ and $t_1,..., t_k \in T_\Omega$, $m(\mathrm{root} = \sigma)(t) = (\delta = \sigma)$, $m(\mathrm{sel}_i)(t) = t_i$ if $i \leqslant k$, and undefined otherwise, and $m(\Sigma) = \lambda t \in T_\Sigma . t$.

Recall that the concept of $X(S)$-transducer is also defined for $X = \mathrm{BT}$.

3.3. DEFINITION. A *basic tree transducer* is a BT(TR)-transducer.

If $M$ is a basic tree transducer with encoding $\Sigma$, then $\Sigma$ is called the *input alphabet* of $M$. Obviously, for every basic tree transducer we can assume that the

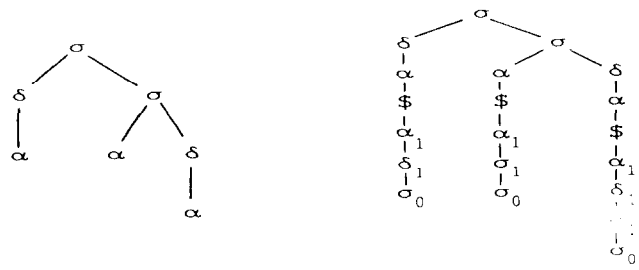

FIG. 1. Input tree and output tree of a basic tree transducer.

boolean expression occurring in the rules are of the form root $= \sigma$ with $\sigma \in \Sigma$ (cf. Lemma 3.18 of [EV]).

In the next example we construct a basic tree transducer, which appends at every leaf lf of a given input tree a monadic tree $t$(lf). If $t$(lf) is viewed as a string, then it is the reverse of the sequence of labels, that is obtained by walking from the root of the input tree to the leaf lf (cf. Fig. 1).

3.4. EXAMPLE. Let $M = (N, \Sigma, \Delta, A_{\text{in}}, R)$ be the basic tree transducer defined by

$$N = \{A_{\text{in}}^{(0)}, A^{(1)}\},$$
$$\Sigma = \{\sigma^{(2)}, \delta^{(1)}, \alpha^{(0)}\},$$
$$\Delta = \{\sigma^{(2)}, \delta^{(1)}, \alpha^{(1)}, \$^{(1)}\} \cup \Delta',$$

where

$$\Delta' = \{\sigma_0^{(0)}, \sigma_1^{(1)}, \delta_0^{(0)}, \delta_1^{(1)}, \alpha_0^{(0)}, \alpha_1^{(1)}\},$$

and $R$ contains the following rules:

$$A_{\text{in}} \to \textit{if root} = \sigma \textit{ then } \sigma(A\langle \text{sel}_1 \rangle(\sigma_0), A\langle \text{sel}_2 \rangle(\sigma_0)),$$
$$A_{\text{in}} \to \textit{if root} = \delta \textit{ then } \delta(A\langle \text{sel}_1 \rangle(\delta_0)),$$
$$A_{\text{in}} \to \textit{if root} = \alpha \textit{ then } \alpha(\$(\alpha_0)),$$
$$A(y) \to \textit{if root} = \sigma \textit{ then } \sigma(A\langle \text{sel}_1 \rangle(\sigma_1(y)), A\langle \text{sel}_2 \rangle(\sigma_1(y))),$$
$$A(y) \to \textit{if root} = \delta \textit{ then } \delta(A\langle \text{sel}_1 \rangle(\delta_1(y))), \text{ and}$$
$$A(y) \to \textit{if root} = \alpha \textit{ then } \alpha(\$(\alpha_1(y))).$$

Note that $M$ is deterministic. Note also that we do not write the syntactic parameter $x$ as we did in the Introduction. On the input tree $s = \sigma(\delta(\alpha), \sigma(\alpha, \delta(\alpha)))$ $M$ computes as follows. (We use $s'$ to abbreviate $\sigma(\alpha, \delta(\alpha))$.)

$$\begin{aligned} A_{\text{in}}\langle s \rangle &\Rightarrow \sigma(A\langle \delta(\alpha) \rangle(\sigma_0), A\langle s' \rangle(\sigma_0)) \\ &\Rightarrow^2 \sigma(\delta(A\langle \alpha \rangle(\delta_1(\sigma_0))), \sigma(A\langle \alpha \rangle(\sigma_1(\sigma_0)), A\langle \delta(\alpha) \rangle(\sigma_1(\sigma_0)))) \\ &\Rightarrow^3 \sigma(\delta(\alpha(t_1)), \sigma(\alpha(t_2), \delta(A\langle \alpha \rangle(\delta_1(\sigma_1(\sigma_0)))))) \end{aligned}$$

(where $t_1 = \$(\alpha_1(\delta_1(\sigma_0)))$ and $t_2 = \$(\alpha_1(\sigma_1(\sigma_0)))$)

$$\Rightarrow \sigma(\delta(\alpha(t_1)), \sigma(\alpha(t_2), \delta(\alpha(t_3))))$$

(where $t_3 = \$(\alpha_1(\delta_1(\sigma_1(\sigma_0))))$).

Figure 1 shows $s$ and the corresponding output tree. For the description of the translation induced by $M$, we define the string homomorphism $h: p(\Sigma)^* \cup p(\Delta)^* \to \Sigma$, which deletes from every symbol in $p(\Sigma)$, the second component, and from every symbol in $p(\Delta)$, the second component and the index. Formally, for every $x \in \{\sigma, \delta, \alpha, \$\}$ and $j \in \{0, 1, 2\}$, $h((x, j)) = x$ and, for every $x \in \{\sigma, \delta, \alpha\}$ and $i, j \in \{0, 1\}$, $h((x_i, j)) = x$. Then $\tau(M) = \{(s, t) \in T_\Sigma \times T_\Delta \mid s = t[\alpha(\$(t')) \leftarrow \alpha; t' \in T_\Delta]$ and $h(\pi_\Delta(t)) = \{w\$w^R \mid w \in h(\pi_\Sigma(s))\}\}$, where $w^R$ is the reversed string of $w$. Obviously, $M$ is total; hence, $\tau(M) \in D_t\mathrm{BT(TR)}$. Note that $h(\pi_\Delta(\tau(M)(T_\Sigma))) = \{w\$w^R \mid w \in h(\pi_\Sigma(s))$ for some $s \in T_\Sigma\} = \{w\$w^R \mid w \in \{\sigma, \delta\}^* \alpha\}$ is in $\mathscr{L}_{\mathrm{LIN}}$, but not in $\mathscr{L}_{\mathrm{REG}}$.

We note that CFT(TR)-transducers and RT(TR)-transducers are very close to OI-macro tree transducers and top-down tree transducers, respectively. In fact, the corresponding classes of induced translations are the same (cf. Theorem 3.19 and Corollary 3.20 of [EV]). Thus on the one hand, basic tree transducers are restricted macro tree transducers and, on the other hand, they are an extension of top-down tree transducers. This also holds for arbitrary storage types rather than only for TR.

3.5. FACT. $\mathrm{RT}(S) \subseteq \mathrm{BT}(S) \subseteq \mathrm{CFT}(S)$ *and determinism and totality are preserved.*

*Proof.* Follows immediately from $\mathrm{RT} \subseteq \mathrm{BT} \subseteq \mathrm{CFT}$. ∎

Since the class of domains of top-down tree transducers and of macro tree transducers is the class of recognizable tree languages, the same property holds for the class of domains of basic tree transducers.

3.6. FACT. $\mathrm{dom(BT(TR))} = \mathrm{RECOG}$.

*Proof.* Follows from Fact 3.5, Theorem 7.4 of [EngVog1] ($\mathrm{dom(CFT(TR))} \subseteq \mathrm{RECOG}$), and the fact that $\mathrm{RECOG} \subseteq \mathrm{dom(RT(TR))}$ [Rou2]. ∎

By using a standard technique, we prove that ranges of basic tree transducers and images of RECOG under basic tree transducers form the same class of tree languages.

3.7. LEMMA. *For every* $n \geq 0$, $\mathrm{range(BT(TR)}^n) = \mathrm{BT(TR)}^n\mathrm{(RECOG)}$.

*Proof.* It clearly suffices to prove the equality for $n = 1$. Since for every ranked alphabet $\Sigma$, $T_\Sigma$ is a recognizable tree language, $\mathrm{range(BT(TR))} \subseteq \mathrm{BT(TR)(RECOG)}$ is trivially true. The other inclusion can be proved in the same

way as $CF(TR)(RECOG) \subseteq range(CF(TR))$ is proved in Theorem 3.2.1 of [EngRozSlu]. ▮

In Example 3.4 comparing the height of an input tree $s$ with the height of the corresponding output tree $t$, one realizes that there is a linear relationship: $height(t) = 2 \cdot height(s) + 1$. In general, a linear relationship holds between the heights of input and output trees. This is due to the fact that on every path through a sentential form at most one nonterminal occurs. Hence, the height of an output tree is bounded by the product of the height of the input tree and the maximal amount, by which every path of a sentential form can be increased in one derivation step.

3.8. LEMMA. *Let $n \geqslant 1$ and let $M_1, M_2, ..., M_n$ be basic tree transducers. Then there is a constant $c \geqslant 0$ such that, for every $(s, t) \in \tau(M_1) \circ \cdots \circ \tau(M_n)$, $height(t) \leqslant c \cdot height(s)$.*

*Proof.* Obviously, it is sufficient to show the existence of such a constant for $n = 1$. Then, for a composition of basic tree transducers $M_1, ..., M_n$, the desired constant is the product of the constants associated with the $M_i$'s. Let $M = (N, \Sigma, \Delta, A_{in}, R)$ be a basic tree transducer. Define $c = \max\{height(\zeta) \mid \zeta$ is a right-hand side of a rule of $R\}$. Then *Claim* 1 can be proved by induction on the height of $s$.

*Claim* 1. For every $A \in N_k$ with $k \geqslant 0$, $s \in T_\Sigma$, $t, t_1, ..., t_k \in T_\Delta$ if $A\langle s\rangle(t_1, ..., t_k) \Rightarrow_M^* t$, then $height(t) \leqslant c \cdot height(s) + \max\{height(t_i) \mid i \in [k]\}$.

The formal proof is left to reader. ▮

The previous lemma shows that arguments on the height of output trees cannot be used in the proof of the strictness of the composition hierarchy of basic tree transducers. (Note that height arguments were used to prove the macro tree transducer hierarchy to be strict, cf. Theorem 4.16 of [EngVog1].) In Chapter 5 we will use arguments on paths to prove this result.

In Example 4.3 of [EngVog1] a total deterministic macro tree transducer is defined, which induces the translation $\{(f^k a, f^{\exp(k)} a) \mid k \geqslant 0\}$, where $f^k a$ denotes the monadic tree with $k$ $f$'s (of rank 1) and $\exp(k) = 2^k$. By Lemma 3.8, this translation cannot be realized by compositions of basic tree transducers.

3.9. COROLLARY. $D_t CFT(TR) - \bigcup \{BT(TR)^n \mid n \geqslant 0\} \neq \varnothing$.

In particular, basic tree transducers are less powerful than macro tree transducers.

3.10 THEOREM. $BT(TR) \subsetneq CFT(TR)$ *and* $D_t BT(TR) \subsetneq D_t CFT(TR)$.

*Proof.* Immediate from Fact 3.5 and Corollary 3.9. ▮

After having studied the tree-to-path translations of basic tree transducers, we will generalize this result to ranges of (compositions of) transducers, cf. Theorem

5.12. Moreover, then we can prove that, on the other hand, basic tree transducers are more powerful than (compositions of) top-down tree transducers, cf. Theorem 5.17.

## 4. Regular Machine Characterization

In this section we implement basic tree transducers on regular machines which may use an additional storage type. Moreover, for two extended versions of basic tree transducers, we even prove characterizations in terms of regular machines. Since it has turned out that the proofs of these characterization results do not depend on any property of the tree storage type, we prove the results for extended BT($S$)-transducers, where $S$ is an arbitrary storage type.

The two considered extensions of the concept of BT($S$)-transducer, viz., regular extended BT($S$)-transducers and finite extended BT($S$)-transducers, are special cases of extended CFT($S$)-transducers (cf. Definition 5.22 of [EV]).

4.1. Definition. An *extended* CFT($S$)*-transducer* is a CFT($S_{\mathrm{id}}$)-transducer $M$, where the rules are restricted as follows. If $A(y_1,\ldots,y_k) \to$ *if* $b$ *then* $\zeta$ is a rule of $M$, then the instruction id may only occur in those subtrees of $\zeta$, which have the form $B\langle \mathrm{id}\rangle(y_1,\ldots,y_k)$.

Note that, if $S$ contains an identity, then $\mathrm{CFT}_{\mathrm{ext}}(S) = \mathrm{CFT}(S)$.

In extended BT($S$)-transducers the parameters of nonterminals may hold tree languages as actual values. However, rather than placing such tree languages explicitly in the parameter positions, e.g., in the form of language names (as it was done in [EngSlu2]), we represent them by nonterminals which can generate the languages. Although this involves nesting of nonterminals, this feature is very restrictive. If a nonterminal $B$ occurs nested in the parameter position of a nonterminal, then in the right-hand sides of $B$-rules no nesting is allowed. This suggests a partition of the set of nonterminals into so-called "basic nonterminals" which may contain nested nonterminals in their parameter positions, and so-called "extension nonterminals" which may occur nested and hence are responsible for the extension.

4.2. Definition. Let $U \in \{\mathrm{reg}, \mathrm{fin}\}$. A *U-extended* BT($S$)*-transducer* $M$ is a $\mathrm{CFT}_{\mathrm{ext}}(S)$-transducer $(N, e, \Delta, A_{\mathrm{in}}, R)$ such that

— there is a partition of $N$ into the set $N_\Phi$ of *basic nonterminals* and the set $N_\Psi$ of *extension nonterminals*

— $A_{\mathrm{in}}^{(0)} \in N_\Phi$

— there is a partition of $R$ into the set $R_\Phi$ of *basic rules* and the set $R_\Psi$ of *extension rules*; every basic rule has the form $A(y_1,\ldots,y_k) \to$ *if* $b$ *then* $\zeta$, where $k \geqslant 0$, $A^{(k)} \in N_\Phi$, $b \in \mathrm{BE}(P)$, $\zeta \in T_\Delta(Y_k \cup \Lambda_k \cup Z_k)$,

$$\Lambda_k = \{B\langle f\rangle(\psi_1\langle \mathrm{id}\rangle \tilde{y},\ldots,\psi_r\langle \mathrm{id}\rangle \tilde{y}) \mid r \geqslant 0,\ B^{(r)} \in N_\Phi,\ f \in F,$$
$$\text{and } \psi_1,\ldots,\psi_r \in N_\Psi \text{ of rank } k\},$$

$\tilde{y}$ abbreviates $(y_1,..., y_k)$, and

if $U = \text{reg}$, then $Z_k = \{B\langle \text{id}\rangle \tilde{y} \mid B^{(k)} \in N_\Phi\}$, and
if $U = \text{fin}$, then $Z_k = \varnothing$,

and every extension rule has the form $\psi(y_1,..., y_k) \to \zeta$, where $k \geqslant 0$, $\psi^{(k)} \in N_\Psi$, $\zeta \in T_\Delta(Y_k \cup Z_k)$, and

if $U = \text{reg}$, then $Z_k = \{\phi\langle \text{id}\rangle(y_1,..., y_k) \mid \phi^{(k)} \in N_\Psi\}$, and
if $U = \text{fin}$, then $Z_k = \varnothing$.

In reg-extended BT($S$)-transducers, the extension nonterminals represent *reg*ular tree languages (i.e., languages in $\mathscr{L}_{\text{RT}}$) and in fin-extended BT($S$)-transducers these languages are *fin*ite. An *extended* BT($S$)*-transducer* is either a reg-extended BT($S$)-transducer or a fin-extended BT($S$)-transducer. For $U \in \{\text{reg}, \text{fin}\}$, a $U$-extended BT($S$)-transducer is also called $\text{BT}_U(S)$-transducer. Clearly, the notions of determinism and totality carry over from $\text{CFT}(S_{\text{id}})$-transducers. The class of translations induced by (total deterministic) $\text{BT}_U(S)$-transducers is denoted by $\text{BT}_U(S)$ ($\text{D}_t\text{BT}_U(S)$, respectively).

If not specified otherwise, we will use $A$, $B$, $A'$, $B'$,... as basic nonterminals and $\psi$, $\phi$, $\psi'$, $\phi'$,... as extension nonterminals. This convention allows us to drop troublesome quantifications whenever the situation is clear anyway.

An easy example of a translation in $\text{BT}_{\text{reg}}(\text{TR}) - \text{BT}(\text{TR})$ is obtained by modifying the basic tree transducer of Example 3.4: above every symbol $\sigma$ of an output tree, a tree from a regular tree language is inserted. We leave the formal description to the reader.

We note that $\text{range}(\text{BT}_{\text{reg}}(S_0))$ is equal to the class UltBT(2) of tree languages which are generated by 2-level ultra-basic tree grammars [EngSlu1]. In $k$-level ultra-basic tree grammars, to every nonterminal a number $n$ with $1 \leqslant n \leqslant k$ is associated, which is called its level. A nonterminal $B$ may occur in the parameter position of a nonterminal $A$ only if the level of $B$ is smaller than the level of $A$. Clearly, in this formalism, the basic nonterminals of a $\text{BT}_{\text{reg}}(S_0)$-transducer have level 2 and the extension nonterminals have level 1. Moreover, we note that $\text{range}(y\text{BT}_{\text{fin}}(S_0))$ coincides with the class EB of string languages, which are generated by extended basic macro grammars [Dow, EngSchvLe], i.e., $\text{range}(y\text{BT}_{\text{fin}}(S_0)) = \text{EB}$. Such grammars are basic macro grammars [Fis] in which the parameter positions of every nonterminal may hold a finite set of terminal strings rather than only one string. It follows from Theorem 3.1 of [EngSlu2] that even $\text{range}(y\text{BT}_{\text{reg}}(S_0)) = \text{range}(y\text{BT}_{\text{fin}}(S_0)) = \text{EB}$. We briefly explain this. Since every extension nonterminal represents a regular tree language and since $\text{yield}(\mathscr{L}_{\text{RT}}) = \mathscr{L}_{\text{CF}}$ (cf. Sect. 2.2), it is intuitively clear that $\text{range}(y\text{BT}_{\text{reg}}(S_0))$ can be generated by $\mathscr{L}_{\text{CF}}$-extended basic macro grammars, i.e., by basic macro grammars in which the sets in parameter positions are context-free languages. $\mathscr{L}_{\text{CF}}$ is contained in the class ETOL of languages generated by ETOL systems [Roz] and ETOL is the hyper-algebraic closure of the class $\mathscr{L}_{\text{FIN}}$ of finite languages [RozSal], i.e., $\text{H}(\mathscr{L}_{\text{FIN}}) = \text{ETOL}$. In particular, Theorem 3.1 of [EngSlu2] says that $\text{H}(\mathscr{L}_{\text{FIN}})$-extended basic macro grammars are equivalent to $\mathscr{L}_{\text{FIN}}$-extended basic macro

grammars. Hence, $\text{range}(y\text{BT}_{\text{reg}}(S_0)) \subseteq \text{range}(y\text{BT}_{\text{fin}}(S_0))$. The other inclusion follows from the definition of $\text{BT}_{\text{fin}}(S)$-transducers.

The next lemma brings some order in the defined subclasses of $\text{CFT}(S)$.

4.3. LEMMA. (i) $\text{BT}(S) \subseteq \text{BT}_{\text{fin}}(S) \subseteq \text{BT}_{\text{reg}}(S)$.

(ii) $\text{BT}_{\text{reg}}(S) \subseteq \text{CFT}_{\text{ext}}(S)$ *and* $\text{BT}_{\text{fin}}(S) \subseteq \text{CFT}(S)$.

(iii) *The inclusions in* (i) *and* (ii) *preserve determinism and totality.*

*Proof.* Given a $\text{BT}(S)$-tranducer. By replacing every tree $t$ in a parameter position of a nonterminal by a new extension nonterminal $\psi$ and by adding the extension rule $\psi\tilde{y} \to t$, where $\tilde{y}$ is the appropriate list of parameters, an equivalent $\text{BT}_{\text{fin}}(S)$-transducer is obtained. The second inclusion of (i) and the first inclusion of (ii) hold by definition. Now let $M$ be a $\text{BT}_{\text{fin}}(S)$-transducer and let $\psi$ be an occurrence of an extension nonterminal in a parameter position of a construct $A\langle f \rangle$. Then replace $\psi\langle \text{id} \rangle$ by $\psi\langle f \rangle$ and view $\psi$ as a usual nonterminal of the $\text{CFT}(S)$-transducer. Clearly, $\tau(M) = \tau(M')$. It is easy to see that the mentioned constructions preserve totality and determinism. ∎

In the total deterministic case both extensions do not increase the power of $\text{BT}(S)$-transducers.

4.4. LEMMA. $\text{D}_{\text{t}}\text{BT}_{\text{reg}}(S) = \text{D}_{\text{t}}\text{BT}_{\text{fin}}(S) = \text{D}_{\text{t}}\text{BT}(S)$.

*Proof.* It follows from parts (i) and (iii) of Lemma 4.3 that it is sufficient to prove $\text{D}_{\text{t}}\text{BT}_{\text{reg}}(S) \subseteq \text{D}_{\text{t}}\text{BT}(S)$. To facilitate this proof, we introduce locally to the present lemma a restriction of $\text{BT}_{\text{fin}}(S)$-transducers. A one-extended $\text{BT}(S)$-transducer is a $\text{BT}_{\text{fin}}(S)$-transducer such that for every extension nonterminal $\psi$ there is exactly one $\psi$-rule. The class of induced translations is denoted by $\text{BT}_{\text{one}}(S)$. We first prove that $\text{D}_{\text{t}}\text{BT}_{\text{reg}}(S) \subseteq \text{D}_{\text{t}}\text{BT}_{\text{one}}(S)$ and second that $\text{D}_{\text{t}}\text{BT}_{\text{one}}(S) \subseteq \text{D}_{\text{t}}\text{BT}(S)$.

Let $M = (N, e, \Delta, A_{\text{in}}, R)$ be a total deterministic $\text{BT}_{\text{reg}}(S)$-transducer in standard test form and let $N_\Phi$ and $N_\Psi$ be the sets of basic nonterminals and of extension nonterminals, respectively. We define two classes of regular tree grammars by means of which the undesired constructs like $Z\langle \text{id} \rangle(y_1, \ldots, y_k)$ $(Z \in N)$ in right-hand sides of rules of $M$ can be replaced.

For every $A^{(k)} \in N_\Phi$ and test $b$ occurring in $M$, we define the regular tree grammar $G(A, b) = (N'_\Phi, \Sigma, \tilde{A}, R'_\Phi)$, where $N'_\Phi = \{\tilde{B}^{(0)} \mid B \in N_\Phi\}$, $\Sigma = \Delta \cup Y_k \cup N\langle \tilde{F} \cup \{\text{id}\} \rangle$ with $\tilde{F}$ denoting the finite set of instruction symbols occurring in $M$, and if $B(y_1, \ldots, y_k) \to \textit{if } b \textit{ then } \zeta$ is a basic rule of $R$, then $\tilde{B} \to \tilde{\zeta}$ is in $R'_\Phi$, where $\tilde{\zeta}$ is obtained from $\zeta$ by replacing every construct of the form $D\langle \text{id} \rangle(y_1, \ldots, y_k)$ with $D \in N_\Phi$ by $\tilde{D}$.

For every $\psi^{(k)} \in N_\Psi$ with $k \geqslant 0$, we define the regular tree grammar $G(\psi) = (N'_\Psi, \Delta \cup Y_k, \tilde{\psi}, R'_\Psi)$, where $N'_\Psi = \{\tilde{\phi}^{(0)} \mid \phi \in N_\Psi\}$ and, if $\phi(y_1, \ldots, y_k) \to \zeta$ is an extension rule of $R$, then $\tilde{\phi} \to \tilde{\zeta}$ is in $R'_\Psi$ and $\tilde{\zeta}$ is obtained from $\zeta$ by replacing every construct of the form $v\langle \text{id} \rangle(y_1, \ldots, y_k)$ by $\tilde{v}$.

Note that, for every $A \in N_\Phi$, test $b$ occurring in $M$, and $\psi \in N_\Psi$, the languages $L(G(A, b))$ and $L(G(\psi))$ contain at most one tree, because $M$ is deterministic.

Now construct the one-extended BT($S$)-transducer $M' = (N, e, \Delta, A_{in}, R')$ as follows.

Let $A(y_1,..., y_k) \to$ *if* $b$ *then* $\zeta$ be a basic rule of $R$. If for every construct $B\langle \mathrm{id} \rangle(y_1,..., y_k)$ occurring in $\zeta$, $L(G(B, b)) \neq \varnothing$, then the rule $A(y_1,..., y_k) \to$ *if* $b$ *then* $\zeta'$ is a basic rule in $R'$, where $\zeta'$ is obtained from $\zeta$ by replacing every construct $B\langle \mathrm{id} \rangle(y_1,..., y_k)$ with $B \in N_\Phi$ by the tree in $L(G(B, b))$ (a basic rule of $R$, for which $L(G(B, b)) = \varnothing$ for some $B$ occurring in the form $B\langle \mathrm{id} \rangle(y_1,..., y_k)$ in $\zeta$, can be disregarded).

Let $\psi(y_1,..., y_k) \to \zeta$ be an extension rule of $M$. Then $\psi(y_1,..., y_k) \to t$ is an extension rule of $R'$, where $t = t_\psi$ if $L(G(\psi)) = \{t_\psi\}$, and $t$ is an arbitrary terminal symbol of rank 0 if $L(G(\psi)) = \varnothing$. Clearly, in a successful derivation of $M'$, an extension nonterminal $\psi$, which does not generate anything, will never occur in an outermost position, and hence, will not contribute to the final terminal tree.

It is easy to check that $M'$ is deterministic. Moreover, it is clear that $\tau(M) = \tau(M')$ and hence, $M'$ is total.

Now construct from $M'$ the equivalent total deterministic BT($S$)-transducer $M'' = (N, e, \Delta, A_{in}, R'')$. If $A(y_1,..., y_k) \to$ *if* $b$ *then* $\zeta$ is a basic rule of $M'$, then replace every construct $\psi\langle \mathrm{id} \rangle(y_1,..., y_k)$ occurring in a parameter position of a basic nonterminal in $\zeta$ by the right-hand side of the $\psi$-rule. Note that there is exactly one $\psi$-rule in $M'$. ▮

Now, what are the appropriate storage types $S_1$ and $S_2$ such that $\mathrm{BT}_{reg}(S)$-transducers and $\mathrm{BT}_{fin}(S)$-transducers are characterized by $\mathrm{RT}(S_1)$-transducers and $\mathrm{RT}(S_2)$-transducers, respectively? In Lemma 4.3 we have seen that $\mathrm{BT}_{reg}(S) \subseteq \mathrm{CFT}_{ext}(S)$ and that $\mathrm{BT}_{fin}(S) \subseteq \mathrm{CFT}(S)$. For $\mathrm{CFT}_{ext}(S)$-transducers and $\mathrm{CFT}(S)$-transducers, machine characterizations are proved in [EV], which use pushdown devices as additional storage types. If these pushdown devices are restricted to be one-turn, then machine characterizations for $\mathrm{BT}_{reg}(S)$ and $\mathrm{BT}_{fin}(S)$ are obtained.

One-turn pushdown automata were considered in [GinSpa1] and proved to characterize the class of linear languages. In [Vog1] the concept of one-turn pushdown is defined as a storage type, i.e., if $S$ is a storage type, then the one-turn pushdown of $S$ (for short: $\mathrm{P}_{1t}(S)$) is also a storage type. A configuration of $\mathrm{P}_{1t}(S)$ is a pushdown in which every square contains a usual pushdown symbol and a configuration of $S$. As soon as a pop instruction has been applied to a pushdown, the application of any further push instruction is undefined. We recall the formal definition in our notational framework.

4.5. DEFINITION. The *one-turn pushdown of* $S$, denoted by $\mathrm{P}_{1t}(S)$, is the storage type $(C', P', F', I', E', m')$, where

$C' = (\Gamma \times C)^+ \times \{0, 1\}$ and $\Gamma$ is a fixed, infinite set of pushdown symbols; the bit $i$, called turn number, indicates whether the turn of a $\mathrm{P}_{1t}(S)$-automaton lies in the future ($i = 0$) or in the past ($i = 1$); the top of the pushdown is at the left,

$P' = \{\text{top} = \gamma \mid \gamma \in \Gamma\} \cup \{\text{test}(p) \mid p \in P\} \cup \{\text{turn} = j \mid j \in \{0, 1\}\}$,

$F' = \{\text{push}(\gamma, f) \mid \gamma \in \Gamma \text{ and } f \in F\} \cup \{\text{pop}\} \cup \{\text{stay}(\gamma) \mid \gamma \in \Gamma\} \cup \{\text{stay}\}$,

$I' = I$,

$E' = \{(\gamma, e) \mid \gamma \in \Gamma \text{ and } e \in E\}$, and for $c' = ((\sigma, c)\beta, i)$ with $\sigma \in \Gamma$, $c \in C$, $\beta \in (\Gamma \times C)^*$, and $i \in \{0, 1\}$,

$m'(\text{top} = \gamma)(c') = (\sigma = \gamma)$,

$m'(\text{test}(p))(c') = m(p)(c)$,

$m'(\text{turn} = j)(c') = (i = j)$,

$m'(\text{push}(\gamma, f))(c') = ((\gamma, m(f)(c))(\sigma, c)\beta, 0)$ if $i = 0$ and $m(f)$ is defined on $c$, and undefined otherwise,

$m'(\text{pop})(c') = (\beta, 1)$ if $\beta \neq \lambda$, and undefined otherwise,

$m'(\text{stay}(\gamma))(c') = ((\gamma, c)\beta, i)$,

$m'(\text{stay})(c') = c'$, and for every $u \in I$, $m'((\gamma, e))(u) = ((\gamma, m(e)(u)), 0)$.

By dropping the bits from the configurations of $P_{lt}(S)$ and by disregarding the restriction on the bit in the definition of the meaning of push$(\gamma, f)$, we obtain the storage type *pushdown of* $S$ (for short: $P(S)$)[Gre2, Eng5, Eng6, EV].

$BT_{reg}(S)$-transducers are characterized by $RT(P_{lt}(TR))$-transducers (cf. Theorem 4.10). Now, which pushdown device corresponds to $BT_{fin}(S)$-transducers? Recall that $CFT(S)$-transducers are equivalent to $RT(P_{bex}(S))$-transducers (cf. Theorem 5.14 of [EV]), where $P_{bex}(S)$ is the "bounded excursion pushdown of $S$" storage type (cf. Definition 5.10 of [EV]). In the next definition we impose the one-turn restriction on this storage type and thereby we obtain the one-turn bounded excursion pushdown of $S$, denoted by $P_{lt,bex}(S)$. Actually, $BT_{fin}(S)$-transducers are characterized by $RT(P_{lt,fin}(S))$-transducers (cf. Theorem 4.10). Note that in every square of a configuration of $P_{bex}(S)$ there is a nonnegative integer installed, which indicates how many excursions have been made from this square. An excursion from a square consists either of a push instruction, a stay$(\gamma)$ instruction, or a stay instruction (the latter two are also called trivial excursions). Instead of repeating the bound of the number of excursions in every square (as in Definition 5.10 of [EV]), the bound here occurs only once in every configuration.

4.6. DEFINITION. The *one-turn bounded excursion pushdown of* $S$, denoted by $P_{lt,bex}(S)$, is the storage type $(C', P', F', I', E', m')$, where

$C' = (\Gamma \times C \times \text{Nat})^+ \times \{0, 1\} \times \text{Nat}$; $\Gamma$ and the second component of a configuration have the same meaning as in Definition 4.5; the nonnegative integer in the third component of a configuration bounds the number of allowed excursions from any single square,

$P' = \{\text{top} = \gamma \mid \gamma \in \Gamma\} \cup \{\text{test}(p) \mid p \in P\} \cup \{\text{turn} = j \mid j \in \{0, 1\}\} \cup \{\text{exc} = j \mid j \geqslant 0\}$,

$F' = \{\text{push}(\gamma, f) \mid \gamma \in \Gamma \text{ and } f \in F\} \cup \{\text{pop}\} \cup \{\text{stay}(\gamma) \mid \gamma \in \Gamma\} \cup \{\text{stay}\}$,

$I' = I$,

$E' = \{(\gamma, e, mx) \mid \gamma \in \Gamma, e \in E, \text{ and } mx \geqslant 0\}$, and for every $c' = ((\sigma, c, k)\beta, i, mx)$ with $\sigma \in \Gamma$, $c \in C$, $k$, $mx \geqslant 0$, $\beta \in (\Gamma \times C \times \text{Nat})^*$, and $i \in \{0, 1\}$,

$m'(\text{top} = \gamma)(c') = (\sigma = \gamma)$,

$m'(\text{test}(p))(c') = m(p)(c)$,

$m'(\text{turn} = j)(c') = (i = j)$,

$m'(\text{exc} = j)(c') = (k = j)$,

$m'(\text{push}(\gamma, f))(c') = ((\gamma, m(f)(c), 0)(\sigma, c, k+1)\beta, 0, mx)$ if $i = 0$, $m(f)$ is defined on $c$, and $k + 1 \leqslant mx$, and undefined otherwise,

$m'(\text{pop})(c') = (\beta, 1, mx)$ if $\beta \neq \lambda$, and undefined otherwise,

$m'(\text{stay}(\gamma))(c') = ((\gamma, c, k+1)\beta, i, mx)$ if $k + 1 \leqslant mx$, and undefined otherwise,

$m'(\text{stay})(c') = ((\sigma, c, k+1)\beta, i, mx)$ if $k + 1 \leqslant mx$, and undefined otherwise, and for every $u \in I$, $m'((\gamma, e, mx))(u) = ((\gamma, m(e)(u), 0), 0, mx)$.

We note that the combination of one-turn and bounded excursion is equivalent to the combination of one-turn and locally finite [vLe]. The latter property restricts the number of consecutive stay instructions.

Again we can drop the bits from the configurations of $P_{1t,bex}(S)$ and disregard the restriction imposed by the bit on the definedness of the push instruction. Then we obtain a slight extension of the bounded excursion pushdown of $S$ as it is defined in Definition 5.10 of [EV]; our storage type allows us to test explicitly the actual number of stay instructions which have been applied to a square. However, by the usual trick of keeping this information in the pushdown symbol and by updating it appropriately, it is easy to prove that the two versions are "equivalent" (cf. Lemma 4.13 of [EngVog3] for a similar construction). Thus we feel free to define here the storage type *bounded excursion pushdown of* $S$ as "$P_{1t,bex}(S)$ without the one-turn restriction," and to denote it also by $P_{bex}(S)$.

4.7. DEFINITION. A *one-turn pushdown machine* is either an $RT(P_{1t}(S))$-transducer or an $RT(P_{1t,bex}(S))$-transducer for some $S$.

*Remark.* (i) The storage type $P_{1t}(S)$ has an identity, viz., stay. However, $P_{1t,bex}(S)$ has *no* identity.

(ii) For a boolean expression $b \in BE(P)$, we denote by test(b) the boolean expression $b[p \leftarrow \text{test}(p); p \in P]$ over predicates of $P_{1t}(S)$.

(iii) The storage type operator $P_{1t}$ can be iterated by defining $P^0_{1t}(S) = S$ and for every $n \geqslant 0$, $P^{n+1}_{1t}(S) = P_{1t}(P^n_{1t}(S))$. In the same way $P_{1t,bex}$ can be iterated.

(iv) For $n \geqslant 0$, the storage types $P^n_{1t}(S_0)$ and $P^n_{1t,bex}(S_0)$ are denoted by $P^n_{1t}$ and by $P^n_{1t,bex}$, respectively.

We start our investigation on regular machine characterizations of extended $BT(S)$-transducers by implementing them on one-turn pushdown machines. Given an extended $BT(S)$-transducer $M$. The equivalent one-turn pushdown machine per-

forms a symbolic expansion of the basic nonterminals by writing the adjacent lists of extension nonterminals on the pushdowns. If a symbolic expansion is completed and a value of a parameter is required, then this value can be substituted by popping the pushdown and by symbolically expanding the appropriate extension nonterminals. Note that these expansions do not involve any additional push instructions. Hence, the pushdown machine is one-turn.

We note that this construction is closely related to the simulation of the storage type "extended tree-pushdown of $S$" by $\mathrm{P}(S)$ (cf. Lemma 5.19 of [EV]) and to the simulation of the storage type "tree-pushdown of $S$" by $\mathrm{P}_{\mathrm{bex}}(S)$ (cf. Lemma 5.11 of [EV]).

4.8. LEMMA. $\mathrm{BT}_{\mathrm{reg}}(S) \subseteq \mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S))$ *and* $\mathrm{BT}_{\mathrm{fin}}(S) \subseteq \mathrm{RT}(\mathrm{P}_{\mathrm{lt,bex}}(S))$. *In both inclusions totality and determinism are preserved.*

*Proof.* Let $M = (N, e, \Delta, A_{\mathrm{in}}, R)$ be a $\mathrm{BT}_{\mathrm{reg}}(S)$-transducer. Then there is a partition of $N$ into the set $N_\Phi$ of basic nonterminals and the set $N_\Psi$ of extension nonterminals, and there is a partition of $R$ into the set $R_\Phi$ of basic rules and the set $R_\Psi$ of extension rules. Let $\tilde{F}$ be the finite set of instructions occurring in $R$. Let $v = \max\{k \mid N_k \neq \emptyset\}$. We can assume that $N_\Phi \cup N_\Psi \cup \tilde{\Psi} \subseteq \Gamma$, where $\tilde{\Psi} = \{(\psi_1, \ldots, \psi_r) \mid 0 \leqslant r \leqslant v \text{ and } \psi_i \in N_\Psi \text{ for every } i \in [r]\}$.

Construct the $\mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S))$-transducer $M' = (N', e', \Delta, *, R')$ as follows:

$$N' = \{*\} \cup \{pr_j \mid j \in [v]\} \cup \{[A, f] \mid A \in N_\Phi \text{ and } f \in \tilde{F}\},$$
$$e' = (A_{\mathrm{in}}, e);$$

and $R'$ is determined as follows:

— If $A\tilde{y} \to$ *if* $b$ *then* $\zeta$ is in $R_\Phi$, where $A^{(k)} \in N_\Phi$, $k \geqslant 0$, and $\tilde{y}$ abbreviates $(y_1, \ldots, y_k)$, then $* \to$ *if* $\mathrm{top} = A$ *and* $\mathrm{test}(b)$ *then* $\zeta'$ is in $R'$, where $\zeta'$ is obtained from $\zeta$ by replacing every

$$B\langle f \rangle(\psi_1\langle \mathrm{id} \rangle \tilde{y}, \ldots, \psi_r\langle \mathrm{id} \rangle \tilde{y}) \quad \text{by } [B, f]\langle \mathrm{stay}((\psi_1, \ldots, \psi_r)) \rangle,$$
$$B\langle \mathrm{id} \rangle \tilde{y} \quad \text{by } *\langle \mathrm{stay}(B) \rangle,$$
$$y_j \quad \text{by } pr_j\langle \mathrm{pop} \rangle.$$

— For every $[B, f] \in N'$, $[B, f] \to *\langle \mathrm{push}(B, f) \rangle$ is in $R'$.

— For every $pr_j \in N'$ and $(\psi_1, \ldots, \psi_r) \in \tilde{\Psi}$ with $j \leqslant r$, $pr_j \to$ *if* $\mathrm{top} = (\psi_1, \ldots, \psi_r)$ *then* $*\langle \mathrm{stay}(\psi_j) \rangle$ is in $R'$.

— If $\psi(y_1, \ldots, y_k) \to \zeta$ is in $R_\Psi$, then $* \to$ *if* $\mathrm{top} = \psi$ *then* $\zeta'$ is in $R'$, where $\zeta'$ is obtained from $\zeta$ by replacing every

$$\phi\langle \mathrm{id} \rangle(y_1, \ldots, y_k) \quad \text{by } *\langle \mathrm{stay}(\phi) \rangle$$

and

$$y_j \quad \text{by } pr_j\langle \mathrm{pop} \rangle.$$

Note that, if $M$ is deterministic, then $M'$ is deterministic. From the following two claims immediately $\tau(M) = \tau(M')$ follows; since $\mathrm{dom}(e) = \mathrm{dom}(e')$, also immediately $\tau(M) = \tau(M')$ follows; since $\mathrm{dom}(e) = \mathrm{dom}(e')$, also totality is preserved. The elements of the set $\bar{Y}_k = \{\bar{y}_1, ..., \bar{y}_k\}$ are of rank 0. Recall the definition of $M(\bar{Y}_k)$ from Section 2.4.

*Claim* 2. For every $A \in N_\Phi$ of rank $k \geq 0$, $c \in C$, and every $t \in T_\Delta(\bar{Y}_k)$, $A\langle c\rangle(\bar{y}_1, ..., \bar{y}_k) \Rightarrow^*_{M(\bar{Y}_k)} t$ iff for every $\beta$, $*\langle((A, c)\beta, 0)\rangle \Rightarrow^*_{M'} t[\bar{y}_j \leftarrow pr_j\langle(\beta, 1)\rangle$; $j \in [k]]$ and $\beta$ is not tested in this derivation.

*Claim* 3. For every $\psi \in N_\Psi$ of rank $k \geq 0$, $c \in C$, and every $t \in T_\Delta(\bar{Y}_k)$, $\psi\langle c\rangle(\bar{y}_1, ..., \bar{y}_k) \Rightarrow^*_{M(\bar{Y}_k)} t$ iff for every $\beta$, $*\langle((\psi, c)\beta, 1)\rangle \Rightarrow^*_{M'} t[\bar{y}_j \leftarrow pr_j\langle(\beta, 1)\rangle$; $j \in [k]]$ and $\beta$ is not tested in this derivation.

The formal proof is left to the reader. If, in particular, $M$ is a $\mathrm{BT}_{\mathrm{fin}}(S)$-transducer, then define $e' = (A_{\mathrm{in}}, e, 3)$. The three excursions are used to write a sequence $(\psi_1, ..., \psi_r)$ on the pushdown, to push, and to replace $(\psi_1, ..., \psi_r)$ by $\psi_j$ for appropriate $j$. Now $M'$ is a $\mathrm{RT}(\mathrm{P}_{\mathrm{lt,bex}}(S))$-transducer and again determinism is preserved. Again $\tau(M) = \tau(M')$ follows from two analogous claims. ∎

Now let us turn to the simulation of an $\mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S))$-transducer $M$ by a reg-extended $\mathrm{BT}(S)$-transducer $M'$. Assume that $M$ is in standard test form, i.e., all tests occurring in $M$ are mutually exclusive (recall the definition from Sect. 2.4). Assume that $M$ has the nonterminals $A_1, ..., A_r$. Then a construct $A\langle((\gamma, c)\beta, 0)\rangle$ ($A$ is a nonterminal, $\gamma$ is a pushdown symbol, $c$ is a configuration of $S$, and $\beta$ is a pushdown configuration) occurring in a sentential form of $M$ is represented by the tree $[A, \gamma]\langle c\rangle(t_1, ..., t_r)$, where $[A, \gamma]$ is a nonterminal and $t_j$ represents the construct $A_j\langle(\beta, 1)\rangle$.

Similarly, $A\langle((\gamma, c)\beta, 1)\rangle$ is represented by $[A, \gamma, b]\langle c\rangle(t_1, ..., t_r)$, where $b$ is that boolean expression over predicates of $S$, which occurs in $M$ and which is true on $c$ (note that this expression is uniquely determined, because $M$ is in standard test form). The reason for having a representation of $\beta$ for every nonterminal $A_i$ of $M$ is the following. If a push instruction is applied to the pushdown $\beta$, then $M$ does not know in advance with which nonterminal it will return to $\beta$. Hence, for every such return nonterminal, $\beta$ has to be represented. The trick of coding finite information by preparing sufficiently many representations of an object (in the parameter positions of nonterminals) was first used in Theorem 7 of [Rou2] to prove that every creative dendrolanguage can be generated by a one-state creative dendrogrammar. Note that creative dendrogrammars generate the class of context-free tree languages. This "Rounds-like trick" occurs at several other places in the literature, cf. e.g., Theorem 1 of [Gue], Construction 1 of [DamGue], Lemma 5.4 of [EV], and Lemma 5.10 of [EngVog3].

In the simulation of one-turn pushdown machines, the extension of basic tree transducers is essential. By constructs like $B\langle\mathrm{id}\rangle(y_1, ..., y_k)$ in basic rules and constructs like $\psi\langle\mathrm{id}\rangle(y_1, ..., y_k)$ in extension rules, stay instructions of an $\mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S))$-transducer are modelled, which occur before the turn of a pushdown and after its

turn, respectively. If the pushdowns are a bounded excursion, then these constructs are not necessary, because a sequence of consecutive stay instructions can be collected and represented by one tree. However, still the finite extension is needed, because the $\mathrm{RT}(\mathrm{P}_{\mathrm{lt,bex}}(S))$-transducer can behave nondeterministically after the turn of one of its pushdowns. The finitely many possibilities are coded by the simulating $\mathrm{BT}_{\mathrm{fin}}(S)$-transducer by means of extension nonterminals.

4.9. LEMMA. $\mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S)) \subseteq \mathrm{BT}_{\mathrm{reg}}(S)$ *and* $\mathrm{RT}(\mathrm{P}_{\mathrm{lt,bex}}(S)) \subseteq \mathrm{BT}_{\mathrm{fin}}(S)$. *In both inclusions determinism and totality are preserved.*

*Proof.* Let $M = (N, e, \Delta, A_1, R)$ be an $\mathrm{RT}(\mathrm{P}_{\mathrm{lt}}(S))$-transducer in standard test form. We can assume that the tests of the rules are of the form top $= \gamma$ *and* test($b$) *and* turn $= i$ for some $\gamma \in \Gamma$, $b \in BE(P)$, and $i \in \{0, 1\}$. (Note that the $b$'s, which occur in these tests, are mutually exclusive.) This allows us to replace every stay instruction in the right-hand side of a rule by a stay($\gamma$) instruction for appropriate $\gamma$. Let $\tilde{\Gamma}$ be the finite set of pushdown symbols, which occur in $M$. Let $N = \{A_1, \ldots, A_r\}$ for some $r \geq 1$ and let $e = (\gamma_0, e')$ for some $\gamma_0 \in \tilde{\Gamma}$ and some encoding symbol $e'$ of $S$.

Construct the $\mathrm{BT}_{\mathrm{reg}}(S)$-transducer $M' = (N', e', \Delta, [A_1, \gamma_0]_0, R')$, where

$N' = N'_\Phi \cup N'_\Psi$ with $N'_\Phi = \{[A, \gamma]_0^{(0)} \mid A \in N \text{ and } \gamma \in \tilde{\Gamma}\} \cup \{[A, \gamma]^{(r)} \mid A \in N \text{ and } \gamma \in \tilde{\Gamma}\}$ and $N'_\Psi = \{[A, \gamma, b]_0^{(0)} \mid A \in N, \gamma \in \tilde{\Gamma}, \text{ and } b \text{ is a test occurring in } R\} \cup \{[A, \gamma, b]^{(r)} \mid A \in N, \gamma \in \tilde{\Gamma}, \text{ and } b \text{ is a test occurring in } R\}$. The nonterminals in $N'_\Phi$ and $N'_\Psi$ are the basic nonterminals and the extension nonterminals, respectively.

$R'$ is determined by the set $R'_\Phi$ of basic rules and the set $R'_\Psi$ of extension rules. In the construction $\tilde{y}$ abbreviates $(y_1, \ldots, y_r)$.

*Construction of* $R'_\Phi$. If $A \to$ *if* top $= \gamma$ *and* test(b) *and* turn $= 0$ *then* $\zeta$ is in $R$, then $[A, \gamma]\, \tilde{y} \to$ *if* $b$ *then* $\zeta'$ is in $R'_\Phi$ and $\zeta'$ is obtained from $\zeta$ by replacing every

$A_j\langle \mathrm{push}(\delta, f)\rangle$ by $[A_j, \delta]\langle f\rangle([A_1, \gamma, b]\langle \mathrm{id}\rangle\, \tilde{y}, \ldots, [A_r, \gamma, b]\langle \mathrm{id}\rangle\, \tilde{y})$,

$A_j\langle \mathrm{pop}\rangle$ by $y_j$,

$A_j\langle \mathrm{stay}(\delta)\rangle$ by $[A_j, \delta]\langle \mathrm{id}\rangle\, \tilde{y}$.

If $\zeta$ does not contain constructs of the form $A_j\langle \mathrm{pop}\rangle$, then, additionally, $R'_\Phi$ contains the rule $[A, \gamma]_0 \to$ *if* $b$ *then* $\zeta'$ and $\zeta'$ is obtained as before except that the constructs $[A_j, \gamma, b]\langle \mathrm{id}\rangle\, \tilde{y}$ and $[A_j, \delta]\langle \mathrm{id}\rangle\, \tilde{y}$, which occur in the substitutes, are replaced by $[A_j, \gamma, b]_0\, \langle \mathrm{id}\rangle$ and by $[A_j, \delta]_0\, \langle \mathrm{id}\rangle$, respectively.

*Construction of* $R'_\Psi$. If $A \to$ *if* top $= \gamma$ *and* test(b) *and* turn $= 1$ *then* $\zeta$ is in $R$ and if $\zeta$ does not contain push instructions, then $[A, \gamma, b]\, \tilde{y} \to \zeta'$ is in $R'_\Psi$ and $\zeta'$ is obtained from $\zeta$ by replacing every

$A_j\langle \mathrm{pop}\rangle$ by $y_j$,

$A_j\langle \mathrm{stay}(\delta)\rangle$ by $[A_j, \delta, b]\langle \mathrm{id}\rangle\, \tilde{y}$.

If $\zeta$ does not contain constructs of the form $A_j\langle \text{pop} \rangle$, then, additionally, $R'_\psi$ contains the rule $[A, \gamma, b]_0 \to \zeta'$, where $\zeta'$ is obtained from $\zeta$ by replacing every $A_j\langle \text{stay}(\delta) \rangle$ by $[A_j, \delta, b]_0 \langle \text{id} \rangle$.

Obviously, if $M$ is deterministic, then $M'$ is deterministic. We leave the formal proof of $\tau(M) = \tau(M')$ to the reader. Note that from this equality the preservation of totality follows.

Now let $M$ be an $\text{RT}(\text{P}_{\text{lt,bex}}(S))$-transducer in standard test form where $e = (\gamma_0, e', mx)$ for some $mx \geqslant 0$, and $\gamma_0$ and $e'$ are as above. Then we can assume that the tests of the rules are of the form $\text{top} = \gamma$ *and* $\text{test}(b)$ *and* $\text{turn} = i$ and $\text{exc} = v$ for some $\gamma \in \Gamma$, $b \in BE(\text{P})$, $i \in \{0, 1\}$, and $0 \leqslant v \leqslant mx$. Furthermore, we can assume that there are no stay instructions in $R$.

First we construct an $\text{RT}(\text{P}_{\text{lt,bex}}(S))$-transducer $M'$, which is equivalent to $M$ and for which the right-hand sides of rules do not contain constructs of the form $A_j\langle \text{stay}(\alpha) \rangle$, but may contain constructs of the form $A_j\langle \text{stay}(\delta_1);\ldots; \text{stay}(\delta_k); \text{push}(\sigma, f) \rangle$ with $\kappa \leqslant mx$ consecutive stay instructions. Then a construction, which is similar to the one given in the first part of this proof, can be applied to $M'$ and the result is an equivalent $\text{BT}_{\text{fin}}(S)$-transducer $M''$. The modification of $M$ is obtained by a collection of regular tree grammars.

For every $A \in N$, $\gamma \in \tilde{\Gamma}$, test $b$ such that $\text{test}(b)$ occurs in $M$, $i \in \{0, 1\}$, and $v$ with $0 \leqslant v \leqslant mx$, the regular tree grammar $G(A, \gamma, b, i, v) = (\tilde{N}, \tilde{\Delta}, [A, v+1, \text{stay}(\gamma)], \tilde{R})$ is specified by

$$\tilde{N} = \{[B, \mu, w] \mid B \in N, \quad v+1 \leqslant \mu \leqslant mx, \quad w = \text{stay}(\delta_1);\ldots; \text{stay}(\delta_\kappa) \text{ with } 1 \leqslant \kappa \leqslant mx \text{ and } \delta_1, \ldots, \delta_\kappa, \sigma \in \tilde{\Gamma}\},$$

$$\tilde{\Delta} = \Delta \cup \{A_j\langle g \rangle^{(0)} \mid A_j \in N \text{ and either } g = \text{pop} \text{ or } g = \text{stay}(\delta_1);\ldots; \text{stay}(\delta_\kappa); \text{push}(\sigma, f) \text{ for some } \kappa \geqslant 1, \delta_1, \ldots, \delta_\kappa \in \tilde{\Gamma} \text{ and } f \in \tilde{F}\},$$

where $\tilde{F}$ is the finite set of instructions occurring in $M$, and $\tilde{R}$ contains the following rules.

If $B \to$ *if* $\text{top} = \delta$ *and* $\text{test}(b)$ *and* $\text{turn} = i$ *and* $\text{exc} = \mu$ *then* $\zeta$ is in $R$, then for every $w$, $[B, \mu, w; \text{stay}(\delta)] \to \zeta'$ is in $\tilde{R}$ and $\zeta'$ is obtained from $\zeta$ by replacing every $A_j\langle \text{stay}(\sigma) \rangle$ by $[A_j, \mu+1, w; \text{stay}(\delta); \text{stay}(\sigma)]$ and every $A_j\langle \text{push}(\sigma, f) \rangle$ by $A_j\langle w; \text{stay}(\delta); \text{push}(\sigma, f) \rangle$.

Note that $L(G(A, \gamma, b, i, v))$ is a finite tree language and, if $M$ is deterministic, then it contains at most one tree.

Now we construct the $\text{RT}(\text{P}_{\text{lt,bex}}(S))$-transducer $M' = (N, e, \Delta, A_1, R')$ which is equivalent to $M$.

If $A \to$ *if* $\text{top} = \gamma$ *and* $\text{test}(b)$ *and* $\text{turn} = i$ *and* $\text{exc} = v$ *then* $\zeta$ is in $R$ and if for every construct $A_j\langle \text{stay}(\delta) \rangle$ occurring in $\zeta$,

$$L(G(A_j, \delta, b, i, v)) \neq \varnothing,$$

then $A \to$ *if* $\text{top} = \gamma$ *and* $\text{test}(b)$ *and* $\text{turn} = i$ *and* $\text{exc} = v$ *then* $\zeta'$ is in $R'$, where $\zeta'$ is obtained from $\zeta$ by replacing every $A_j\langle \text{stay}(\delta) \rangle$ by a tree in $L(G(A_j, \delta, b, i, v))$.

Obviously, $\tau(M) = \tau(M')$ and if $M$ is deterministic, then so is $M'$. Hence, also totality is preserved.

Finally, we construct the $\mathrm{BT}_{\mathrm{fin}}(S)$-transducer $M''=(N'', e', \Delta, [A_1, \gamma_0]_0, R'')$, where $N''=N''_\Phi \cup N''_\Psi$ and $N''_\Phi=\{[A,\gamma]_0^{(0)} \mid A\in N \text{ and } \gamma\in\tilde{\Gamma}\}\cup\{[A,\gamma]^{(r)} \mid A\in N \text{ and } \gamma\in\tilde{\Gamma}\}$,
$N''_\Psi=\{[A,\gamma,b,v]_0^{(0)} \mid A\in N,\ \gamma\in\tilde{\Gamma},\ b \text{ is a test occurring in } M, \text{ and } 0\leqslant v\leqslant mx\}\cup$
$\{[A,\gamma,b,v]^{(r)} \mid A\in N,\ \gamma\in\tilde{\Gamma},\ b \text{ is a test occurring in } M, \text{ and } 0\leqslant v\leqslant mx\}$
and $R''=R''_\Phi\cup R''_\Psi$.

*Construction of* $R''_\Phi$. If $A\to$ *if* top $=\gamma$ *and* test($b$) *and* turn $=0$ *and* exc $=v$ *then* $\zeta$ is in $R'$, then $[A,\gamma]\,\tilde{y}\to\zeta'$ is in $R''_\Phi$, where $\tilde{y}$ abbreviates $(y_1,\ldots,y_r)$ and $\zeta'$ is obtained from $\zeta$ by replacing every

$A_j\langle\mathrm{push}(\sigma,f)\rangle$ by $[A_j,\sigma]\langle f\rangle([A_1,\gamma,b,v+1]\langle\mathrm{id}\rangle\tilde{y},\ldots,[A_r,\gamma,b,v+1]\langle\mathrm{id}\rangle\tilde{y})$,

$A_j\langle\mathrm{stay}(\delta_1);\ldots;\mathrm{stay}(\delta_k);\ \mathrm{push}(\sigma,f)\rangle$ by $[A_j,\sigma]\langle f\rangle([A_1,\delta_\kappa,b,v+\kappa+1]\langle\mathrm{id}\rangle\tilde{y},\ldots,[A_r,\delta_\kappa,b,v+\kappa+1]\langle\mathrm{id}\rangle\tilde{y})$, and

$A_j\langle\mathrm{pop}\rangle$ by $y_j$.

If $\zeta$ does not contain constructs of the form $A_j\langle\mathrm{pop}\rangle$, then, additionally, $R''_\Phi$ contains the rule $[A,\gamma]_0\to\zeta'$ and $\zeta'$ is obtained as before except that the constructs $[A,\gamma,b,\mu]\langle\mathrm{id}\rangle\tilde{y}$, which occur in the substitutes, are replaced by $[A,\gamma,b,\mu]_0\langle\mathrm{id}\rangle$.

*Construction of* $R''_\Psi$. If $A\to$ *if* top $=\gamma$ *and* test($b$) *and* turn $=1$ *and* exc $=v$ *then* $\zeta$ is in $R'$, then $[A,\gamma,b,v]\to\zeta'$ is in $R''_\Psi$, where $\zeta'$ is obtained from $\zeta$ by replacing every $A_j\langle\mathrm{pop}\rangle$ by $y_j$.

Although the construction is a bit technical, it is not so difficult to see that determinism is preserved and that $\tau(M')=\tau(M'')$. Hence, also totality is preserved. ▮

From the two previous lemmas we obtain the desired characterization of extended BT($S$)-transducers by one-turn pushdown machines.

4.10. THEOREM. $\mathrm{BT}_{\mathrm{reg}}(S)=\mathrm{RT}(\mathrm{P}_{1\mathrm{t}}(S))$ *and* $\mathrm{BT}_{\mathrm{fin}}(S)=\mathrm{RT}(\mathrm{P}_{1\mathrm{t,bex}}(S))$. *In both equations totality and determinism are preserved:* $\mathrm{D_t BT}(S)=\mathrm{D_t RT}(\mathrm{P}_{1\mathrm{t}}(S))$.

*Proof.* Lemmas 4.8, 4.9, and 4.4. ▮

In particular, $\mathrm{BT}_{\mathrm{reg}}(S_0)=\mathrm{RT}(\mathrm{P}_{1\mathrm{t}})$. Now recall from the discussion after Definition 4.2 that range($\mathrm{BT}_{\mathrm{reg}}(S_0)$) is the class UltBT(2) of 2-level ultra-basic tree languages. Hence, the languages in UltBT(2) are recognized by top-down tree automata with a one-turn pushdown as storage (recall from Sect. 2.4 the connection of range(RT($S'$)) and top-down tree automata with storage $S'$). Note that (OI) context-free tree languages are recognized by top-down tree automata with a pushdown as storage, i.e., CFT = range(RT(P)) (see [Gue], where these automata are called pushdown tree automata).

After Definition 4.2 it was also observed that range($y\mathrm{BT}_{\mathrm{reg}}(S_0)$) is the class EB of languages generated by extended basic macro grammars. By Theorem 4.10, EB = range($y\mathrm{RT}(\mathrm{P}_{1\mathrm{t}})$), and since $y\mathrm{RT}(S)=\mathrm{CF}(S)$ (cf. Sect. 2.4), EB = range(CF($\mathrm{P}_{1\mathrm{t}}$)).

Remembering that range(CF(P)) is the class of indexed languages (cf. [Eng6]), it is obvious, that the class range($CF(P_{lt})$) is generated by restricted indexed grammars [Aho1]; in such grammars nonterminals produced by flag-consuming productions, never introduce new flags. Hence, we reobtain the characterization of EB by restricted indexed grammars (cf. [Fil, EngSchvLe]).

Since, for every $S$ which contains an identity, $CF(S) = REG(P(S))$ (cf. Theorem 5.1 of [Eng6]), EB is also equal to range($REG(P(P_{lt}))) = \mathscr{L}(P(P_{lt}))$, i.e., extended basic macro languages are characterized by one-way $P(P_{lt})$-automata. (Recall that $P_{lt}(S)$ contains an identity.) In [EngSchvLe] the class EB is characterized by stack-pushdown machines (for short: s-pd machines). Actually, it is not so difficult to see that s-pd machines are equivalent to $P(P_{lt})$-automata; this is a special case of the equivalence of nested stack automata and $P^2$-automata (cf. Theorem 7.4 of [EV]). Thus, perhaps, the machine characterization of $BT_{reg}(S)$-transducers helps to understand the equivalence of extended basic macro grammars and s-pd machines.

Obviously, Theorem 4.10 generalizes the equivalence of linear grammars and one-turn pushdown automata [GinSpal] to the tree case: consider a $BT_{reg}(S_0)$-transducer $M$, in which the alphabets of nonterminals and terminals are monadic; then, by glueing every extension nonterminal to the corresponding basic nonterminal, the rules of $M$ turn into rules of a linear grammar.

Finally, we note that also a characterization of $BT(S)$-transducers can be obtained from the constructions of Lemmas 4.8 and 4.9. It can be shown that $BT(S)$ is equal to the subclass of $RT(P_{lt,bex}(S))$, that is induced by one-turn pushdown machines, which are deterministic after the turn. This corresponds to the equivalence of basic macro grammars and stack-deterministic s-pd machines, cf. Section 5 of [EngSchvLe].

Considering in Theorem 4.10 the tree storage type, we obtain the desired characterizations of extended basic tree transducers in terms of one-turn pushdown machines. But in the total deterministic case we can even obtain a characterization of the composition of basic tree transducers in terms of iterated one-turn pushdown machines. This situation is very similar to the characterization of the composition of total deterministic macro tree transducers by iterated pushdown machines, which is proved in Theorem 8.12 of [EV]. Actually, we obtain the characterization of $D_t BT(TR)^n$ along the same line and, in fact, we can take over the crucial Lemmata 8.5 and 8.9 of [EV]. These two lemmata are summarized as follows.

4.11. LEMMA. $D_t BT(S) = D_t RT(S) \circ D_t BT(TR)$.

*Proof.* "$D_t BT(S) \subseteq D_t RT(S) \circ D_t BT(TR)$": This follows immediately as a special case of Lemma 8.5 of [EV], because in a $BT(S)$-transducer the parameter positions of nonterminals in right-hand sides of rules do not contain nonterminals. Hence, trivially, the presence of nested nonterminals is preserved during every derivation. Note that this preservation property was essential in the proof of Lemma 8.5 of [EV].

"$D_t RT(S) \circ D_t BT(TR) \subseteq D_t BT(S)$": This is a special case of Lemma 8.9 of [EV], where $D_t RT(S) \circ D_t CFT(TR) \subseteq D_t CFT(S)$ is proved. ∎

Now we can formally state the characterization of extended basic tree transducers and compositions of total deterministic basic tree transducers in terms of iterated one-turn pushdown machines.

4.12. THEOREM. (i) $BT_{reg}(TR) = RT(P_{1t}(TR))$ *and* $BT_{fin}(TR) = RT(P_{1t,bex}(TR))$.

(ii) *For every* $n \geqslant 1$, $D_t BT(TR)^n = D_t RT(P^n_{1t}(TR))$.

*Proof.* (i) Follows from Theorem 4.10 by substituting TR for $S$.

(ii) Proved by induction on $n$. The case $n = 1$ follows from Theorem 4.10. Assume that $D_t BT(TR)^n = D_t RT(P^n_{1t}(TR))$ holds. Then

$$\begin{aligned} D_t BT(TR)^{n+1} &= D_t BT(TR)^n \circ D_t BT(TR) \\ &= D_t RT(P^n_{1t}(TR)) \circ D_t BT(TR) \qquad \text{by induction hypothesis} \\ &= D_t BT(P^n_{1t}(TR)) \qquad \text{by Lemma 4.11} \\ &= D_t RT(P^{n+1}_{1t}(TR)) \qquad \text{by Theorem 4.10.} \end{aligned}$$ ∎

## 5. PATH LANGUAGES AND COMPOSITION HIERARCHY

In Section 5.1 we study the tree-to-path translations of compositions of basic tree transducers, i.e., the class $BT(TR)^n \circ \pi$ for some $n \geqslant 0$. We prove that it is contained in $RELAB \circ \pi \circ LIN(\text{one-way})^n$ (cf. Theorem 5.8). In Section 5.2 this rather technical result on classes of translations is applied to a particular class of tree languages, viz., RECOG. Thereby we obtain the inclusion $\pi(BT(TR)^n(RECOG)) \subseteq CTRL_n(LIN, \mathscr{L}_{REG})$ (cf. Theorem 5.9), which also forms the basis of the proof of the composition hierarchy of basic tree transducers (cf. Theorem 5.12). In Section 5.3 we show the connections between (compositions of) top-down tree transducers, basic tree transducers, and macro tree transducers by means of path arguments.

### 5.1. *Tree-to-path Translations*

The first step in the study of $BT(TR)^n \circ \pi$ is the treatment of the class $BT(S) \circ \pi$. Clearly, one might expect that it is contained in $LIN(S)$; just take the paths of the right-hand sides of the involved $BT(S)$-transducer as right-hand sides for rules of the $LIN(S)$-transducer. Although this is a well-educated guess, the inclusion does not necessarily hold. To understand this, let us consider a $BT(S)$-transducer $M$ and the ad hoc constructed $LIN(S)$-transducer $M'$. Then, e.g., if $r: A(y_1) \to \textit{if } b \textit{ then } \sigma(A\langle f \rangle(a), B\langle g \rangle)$ is a rule of $M$ ($A$ and $B$ are nonterminals of rank 1 and 0, respectively, $\sigma$ and $a$ are terminals of rank 2 and 0, respectively, $b$ is a boolean expression, and $f$ and $g$ are instructions), then $r_1: A \to \textit{if } b \textit{ then } (\sigma, 1)\, A\langle f \rangle(a, 0)$ and $r_2: A \to \textit{if } b \textit{ then } (\sigma, 2)\, B\langle g \rangle$ might be rules of $M'$ (recall that $(\sigma, 1)$, $(\sigma, 2)$, and

$(a, 0)$ are elements of a path alphabet). Now assume that the nonterminal $B$ blocks every derivation of $M$, e.g., that there is no $B$-rule in $M$. Hence, the application of rule $r$ never leads to a terminal tree. However, it is very well possible that, by means of rule $r_1$, $M'$ computes a terminal string $w$. But now there is no tree in range$(\tau(M))$ of which $w$ is a path. Thus in general, what is yet missing in the construction of rules $r_i$ of $M'$ from a rule $r$ of $M$ is a facility to check whether there is a successful derivation of $M$ starting from those nonterminals of $r$, that are not considered in $r_i$ (in our example this is nonterminal $B$), together with the corresponding configuration of $S$ (i.e., with $m(g)(c)$ if rule $r$ is applied to $A\langle c\rangle$). How can we formalize this facility?

Here the concept of look-ahead on storage types proves again its usefulness. It was introduced in [Eng6] as an operator on storage types. Given a storage type $S$, the storage type $S$ with look-ahead has additional predicates, which can check properties of successors $m(f_n)(...m(f_2)(m(f_1)(c))...)$ (where $f_1, f_2, ..., f_n$ are instructions of $S$) of a configuration $c$ of $S$ without transforming $c$. We recall the formal definition from [EV].

5.1. DEFINITION. The storage type $S$ *with look-ahead*, denoted $S_{\mathrm{LA}}$, is the tuple $(C, P', F, I, E, m')$, where $m'$ restricted to $P \cup F \cup E$ is equal to $m$, $P' = P \cup \{\langle A, H\rangle \mid H \text{ is a CF}(S)\text{-transducer and } A \text{ is one of its nonterminals}\}$, and for every $c \in C$, $m(\langle A, H\rangle)(c) = \text{true}$ iff there is a $w \in \Delta^*$ such that $A\langle c\rangle \Rightarrow_H^* w$, where $\Delta$ is the terminal alphabet of $H$.

A predicate $\langle A, H\rangle$ of $S_{\mathrm{LA}}$ is also called a *look-ahead predicate* of $S$. Taking yields, it is obvious that we may also consider look-ahead predicates $\langle A, H\rangle$, where $H$ is an RT$(S)$-transducer rather than a CF$(S)$-transducer. We also note that RT$(\mathrm{TR}_{\mathrm{LA}})$-transducers and CFT$(\mathrm{TR}_{\mathrm{LA}})$-transducers are equivalent to top-down tree transducers with regular look-ahead [Eng2] and to macro tree transducers with regular look-ahead [EngVog1], respectively.

Now it is not surprising that the inclusion $\mathrm{BT}(S) \circ \pi \subseteq \mathrm{LIN}(S_{LA})$ holds. In one stroke, we also prove the corresponding result for RT$(S)$ instead of BT$(S)$. Recall from Section 2.2 the definition of path alphabet.

5.2. LEMMA. $\mathrm{BT}(S) \circ \pi \subseteq \mathrm{LIN}(S_{\mathrm{LA}})$ *and* $\mathrm{RT}(S) \circ \pi \subseteq \mathrm{REG}(S_{\mathrm{LA}})$.

*Proof.* Let $M = (N, e, \Delta, A_{\mathrm{in}}, R)$ be a BT$(S)$-transducer and let $\pi_\Sigma$ be in $\pi$. We can assume that $\Sigma = \Delta$. Then construct the LIN$(S_{\mathrm{LA}})$-transducer $M' = (N', e, p(\Delta), [A_{\mathrm{in}}, 0], R')$ with $N' = \{[A, i] \mid A \in N_k \text{ with } k \geqslant 0 \text{ and } 0 \leqslant i \leqslant k\}$ and $R'$ is determined as follows. Consider a rule $A(y_1, ..., y_k) \to \textit{if } b \textit{ then } \zeta$ in $R$. Let $(\delta_1, \kappa_1)(\delta_2, \kappa_2) \cdots (\delta_r, \kappa_r)$ be any path of $\zeta$ with $r \geqslant 1$ and $\delta_i \in N\langle F\rangle \cup \Delta \cup Y_k$ for every $i \in [r]$.

— If for some $i \in [r]$, $\delta_i = B\langle f\rangle$ with $B \in N$ and $f \in F$, then $[A, 0] \to \textit{if } b \textit{ and}$ test *then*

$$(\delta_1, \kappa_1)(\delta_2, \kappa_2) \cdots (\delta_{i-1}, \kappa_{i-1})[B, 0]\langle f\rangle$$

and $[A, \eta] \to$ *if* $b$ *and* test *then*

$$(\delta_1, \kappa_1)(\delta_2, \kappa_2)\cdots(\delta_{i-1}, \kappa_{i-1})[B, \kappa_i]\langle f\rangle(\delta_{i+1}, \kappa_{i+1})\cdots(\delta_{r-1}, \kappa_{r-1})\psi$$

are rules in $R'$, where $\eta$, $\psi$, and test are specified as follows.

If $\delta_r \in \Delta$, then $\eta = 0$ and $\psi = (\delta_r, \kappa_r)$ and if $\delta_r = y_i$ for some $i \in [k]$, then $\eta = i$ and $\psi = \lambda$; "test" abbreviates the conjunction of all look-ahead predicates $\langle \tilde{A}, M_{B,f}\rangle$ such that $B\langle f\rangle$ occurs in $\zeta$. For every $B^{(n)} \in N$ with $n \geq 0$ and every $f \in F$, $M_{B,f}$ is the RT($S$)-transducer $(N' \cup \{\tilde{A}\}, e, \Delta \cup Y_v, \tilde{A}, R_{B,f})$, where $N' = \{A^{(0)} \mid A \in N\}$ and $\tilde{A}$ is a new nonterminal of rank 0, $v = \max\{i \mid N_i \neq \varnothing\}$, $\tilde{A} \to B\langle f\rangle$ is in $R_{B,f}$, and if $A(y_1, \ldots, y_k) \to$ *if* $b$ *then* $\zeta$ is in $R$, then $A \to$ *if* $b$ *then* $\zeta'$ is in $R_{B,f}$ and $\zeta'$ is obtained from $\zeta$ by disregarding the trees occurring in parameter positions of nonterminals.

— If for every $i \in [r]$, $\delta_i \in \Delta \cup Y_k$, then $[A, \eta] \to$ *if* $b$ *and* test *then* $(\delta_1, \kappa_1)(\delta_2, \kappa_2)\cdots(\delta_{r-1}, \kappa_{r-1})\psi$ is a rule in $R'$, where $\eta$, $\psi$, and test are defined as above.

If $M$ is an RT($S$)-transducer, then $M'$ is a REG($S_{\mathrm{LA}}$)-transducer. The correctness of the construction, i.e., $\tau(M) \circ \pi_\Delta = \tau(M')$, follows from Claim 4.

*Claim* 4. For every $A \in N$ of rank $k$ with $k \geq 0$, $c \in C$, and $w \in p(\Delta)^*$,

(i) $A\langle c\rangle(\bar{y}_1, \ldots, \bar{y}_k) \Rightarrow^*_{M(\bar{Y}_k)} t$ and $w(\bar{y}_i, 0)$ is a path of $t$ iff $[A, i]\langle c\rangle \Rightarrow^*_{M'} w$ and

(ii) $A\langle c\rangle(\bar{y}_1, \ldots, \bar{y}_k) \Rightarrow^*_{M(\bar{Y}_k)} t$ and $w$ is a path of $t$ iff $[A, 0]\langle c\rangle \Rightarrow^*_{M'} w$. ∎

Note that, apart from look-ahead, this construction is very similar to the one used in [EngSlu1] to prove that path languages of context-free tree languages are context-free languages. The reason why the extension with look-ahead (or at least something similar) is not needed there, is the fact that a context-free tree grammar can be equivalently transformed such that from *every* nonterminal a terminal tree is derivable (cf. also the proof of the second part of Corollary 5.3(i)). In general, this transformation (see Theorem 3.1.5 of [Fis]) is not possible for $X(S)$-transducers.

The second part of the previous lemma is of independent interest, because it provides bounds for the classes of path languages of images under high level tree transducers [EngVog3] and of path languages of high level tree languages [Dam, EngVog 3]. In particular, the classes of images under macro tree transducers and of context-free tree languages are covered by these results. For every $n \geq 0$, $n - T$ denotes the class of $n$-level tree grammars.

5.3. THEOREM. (i) *For every* $n \geq 0$, $(n+1) - T(\mathrm{TR}) \circ \pi \subseteq \mathrm{yield}(n - T(\mathrm{TR}))$ *and* $\pi(\mathscr{L}_{(n+1)-T}) \subseteq \mathrm{yield}(\mathscr{L}_{n-T})$

(ii) $\mathrm{CFT}(\mathrm{TR}) \circ \pi \subseteq \mathrm{CF}(\mathrm{TR})$ *and* $\pi(\mathscr{L}_{\mathrm{CFT}}) \subseteq \mathscr{L}_{\mathrm{CF}}$.

*Proof.* Since $\mathrm{CFT}(S) = 1 - T(S)$ and $\mathrm{CF}(S) = \mathrm{yield}(0 - T(S))$ (Fact 4.6 of [EngVog3]), (ii) follows from (i) with $n = 0$. Now let $S$ be either TR or $S_0$. Then, $(n+1) - T(S) \circ \pi = \mathrm{RT}(\mathrm{P}^{n+1}_{\mathrm{bex}}(S)) \circ \pi$ (by Theorem 6.15 of [EngVog3] and Fact 4.6

of [EngVog3]) $\subseteq$ REG($P_{bex}^{n+1}(S)_{LA}$) (by Lemma 5.2) = REG($P_{bex}^{n+1}(S_{LA})$) (by Lemma 7.6 of [EngVog3], monotonicity of $P_{bex}$ proved in Lemma 4.14 of [EngVog3], and by Theorem 4.18 of [EV]).

By Lemma 7.7 of [EngVog3], monotonicity of $P_{bex}$, and Theorem 4.18 of [EV] it follows that REG($P_{bex}^{n+1}(TR_{LA})$) = REG($P_{bex}^{n+1}(TR)$). Hence, by Theorem 8.6 of [EngVog3], $(n+1) - T(TR) \circ \pi \subseteq$ yield$(n - T(TR))$. This proves the first part of (i). In particular, we proved that $(n+1) - T(S_0) \circ \pi \subseteq$ REG($P_{bex}^{n+1}((S_0)_{LA})$). In Lemma 2.6 of [EngVog4] it was shown that $S_0$-automata are "closed under look-ahead."

With a similar proof, it can be shown that REG($P_{bex}^{n+1}((S_0)_{LA})$) = REG($P_{bex}^{n+1}(S_0)$). Since range$((n+1) - T(S_0)) = \mathscr{L}_{(n+1)-T}$ (by Lemma 4.7 of [EngVog3]), it follows from Theorem 8.6 of [EngVog3] that $\pi(\mathscr{L}_{(n+1)-T}) \subseteq$ yield$(\mathscr{L}_{n-T})$. ∎

Note that the second part of (i) was already proved in [Dam]. Moreover, it was claimed in [Eng3] that $\pi$(CFT(TR)(RECOG)) is contained in CF(TR)(RECOG).

Next we decompose LIN($TR_{LA}$) into RELAB followed by LIN(TR) (recall the definition of RELAB from Sect. 2.2). Consider a look-ahead predicate $\langle A, H \rangle$, where $H$ is a CF(TR)-transducer and $A$ is a nonterminal of $H$, and the language $L = \{t \mid \langle A, H \rangle$ is true on $t\}$. Clearly, $L$ can be obtained as the domain of a CF(TR)-transducer $H'$, which is an easy modification of $H$. Since dom(CF(TR)) = dom(RT(TR)) = RECOG [Rou2], $L$ is a recognizable tree language. This means that every look-ahead predicate of $TR_{LA}$ specifies a recognizable tree language. Now we decompose a LIN($TR_{LA}$)-transducer $M$ into a relabeling followed by a LIN(TR)-transducer, where the relabeling adds to every node of an input tree of $M$ some extra information; at the root of a tree $t$, this information describes whether the subtrees of $t$ are contained in those recognizable tree languages, which are specified by the look-ahead predicates occurring in $M$.

5.4. LEMMA. LIN($TR_{LA}$) $\subseteq$ RELAB $\circ$ LIN(TR) *and* REG($TR_{LA}$) $\subseteq$ RELAB $\circ$ REG(TR).

*Proof.* Both inclusions are special cases of Theorem 2.6 of [Eng2]. There it is proved that RT($TR_{LA}$) $\subseteq$ RELAB $\circ$ RT(TR). Actually, the whole proof carries over. ∎

LIN(TR)-transducers have a very special feature. In every computation of a LIN(TR)-transducer $M$, only one path of a given input tree is considered. Hence, we can also "feed" $M$ just with the paths of an input tree rather than with the tree itself. This allows us to describe LIN(TR) in terms of string-to-string transducers. Recall the definition of the storage type one-way from Section 2.5.

5.5. LEMMA. LIN(TR) $\subseteq \pi \circ$ LIN(one-way) *and* REG(TR) $\subseteq \pi \circ$ REG(one-way).

*Proof.* We only prove the first inclusion, because the second one is a special case of it. Let $M = (N, \Sigma, \Delta, A_{in}, R)$ be a LIN(TR)-transducer. We construct a LIN(one-way)-transducer $M' = (N, p(\Sigma), \Delta, A_{in}, R')$ such that $\tau(M) = \pi_\Sigma \circ \tau(M')$.

— If $A \to$ *if* root $= \sigma$ *then* $uB\langle \mathrm{sel}_i \rangle v$ is in $R$, with $A, B \in N$, $\sigma \in \Sigma$, and $u, v \in \Delta^*$, then $A \to$ *if* sym $= (\sigma, i)$ *then* $uB\langle \mathrm{read} \rangle v$ is in $R'$.

— If $A \to$ *if* root $= \sigma$ *then* $u$ is in $R$, with $A \in N$, $\sigma \in \Sigma_n$, $n \geqslant 0$, and $u \in \Delta^*$, then $A \to$ *if* sym $= (\sigma, n)$ *then* $u$ is in $R'$. ▮

In the next lemma we collect our knowledge about the class $\mathrm{BT(TR)} \circ \pi$.

5.6. LEMMA. $\mathrm{BT(TR)} \circ \pi \subseteq \mathrm{RELAB} \circ \pi \circ \mathrm{LIN(one\text{-}way)}$ *and* $\mathrm{RT(TR)} \circ \pi \subseteq \mathrm{RELAB} \circ \pi \circ \mathrm{REG(one\text{-}way)}$.

*Proof.* Immediate from the previous three lemmas. ▮

Recall that our aim is the description of the class $\mathrm{BT(TR)}^n \circ \pi$. Actually, the connection between basic tree transducers and LIN(one-way)-transducers, as stated in Lemma 5.6, can be generalized to compositions of both classes of translations. In the inductive proof of this connection we will be faced with the composition of BT(TR)-transducers and relabelings. By a direct construction we prove that BT(TR) is closed under right-composition with RELAB, i.e., $\mathrm{BT(TR)} \circ \mathrm{RELAB} \subseteq \mathrm{BT(TR)}$. In the following we describe the construction, which is very similar to the construction used in [Fis] to prove the closure of IO-macro languages under intersection with regular languages.

Let $M_1$ be a basic tree transducer and let $M_2$ be a relabeling (recall that $M_2$ is a total deterministic bottom-up finite state relabeling). We want to show that $\tau(M_1) \circ \tau(M_2)$ can be realized by a basic tree transducer $M$. Consider a rule $A(y_1, \ldots, y_n) \to$ *if* $b$ *then* $\zeta$ of $M_1$. Assume that $M_2$ has translated some actual parameters $t_1, \ldots, t_n$ of $A$ and has arrived in their roots in the states $q_1, \ldots, q_n$, respectively. Roughly speaking, the relabeling is imitated by $M$ via a direct translation of $\zeta$ and a guessing of transition tables of states of $M_2$ at the nonterminals of $\zeta$. More precisely, $\zeta$ is translated via $M_2$ by starting at the leaves and by assuming that every $y_j$ is already "translated" into $q_j$. Then, in particular, for each tree which occurs in a parameter position of a nonterminal $B_i$ in $\zeta$, the translation yields a state of $M_2$. In $M$ all such states, say, $p_{i,1}, \ldots, p_{i,vi}$, are associated with $B_i$. Then $M$ guesses into which state $M_2$ would translate a terminal tree computed from $B_i$ under the assumption that the parameters, which occur in this tree, have been "translated" into the states $p_{i,1}, \ldots, p_{i,vi}$. This state, say, $p_i$, is also associated with $B_i$. Hence, the transition table $(p_{i,1}, \ldots, p_{i,vi}) \to p_i$ is associated with $B_i$. Finally, having guessed these states (for every nonterminal in $\zeta$ one state is guessed), $M_2$ can compute a state $q$ at the root of $\zeta$. Then the transition table $(q_1, \ldots, q_n) \to q$ is associated with the nonterminal $A$ in the left-hand side of the constructed rule.

5.7. LEMMA. $\mathrm{BT(TR)} \circ \mathrm{RELAB} \subseteq \mathrm{BT(TR)}$ *and* $\mathrm{RT(TR)} \circ \mathrm{RELAB} \subseteq \mathrm{RT(TR)}$.

*Proof.* Let $M_1 = (N^1, \Sigma, \Delta, A^1_{\mathrm{in}}, R_1)$ be a basic tree transducer and let $M_2 = (Q, \Delta, A, R_2)$ be a relabeling. Construct the basic tree transducer $M = (N, \Sigma, A, A_{\mathrm{in}}, R)$ by

$$N = \{[A, (q_1,..., q_n) \to q]^{(n)} \mid A^{(n)} \in N^1, n \geqslant 0, q, q_1,..., q_n \in Q\}$$
$$\cup \{A_{\text{in}}^{(0)}\}$$

and $R$ is defined as follows: If $A(y_1,..., y_n) \to$ *if* root $= \sigma$ *then* $\zeta$ is in $R_1$ with $\zeta = t[z_i \leftarrow B_i\langle \text{sel}_{\kappa i}\rangle(\zeta_{i,1},..., \zeta_{i,vi}); i \in [m]]$, where $t \in T_\Delta(Z_m \cup Y)$ with $m \geqslant 0$ and $t$ is linear in $Z_m = \{z_1,..., z_m\}$, for every $i \in [m]$, $vi \geqslant 0$, $B_i^{(vi)} \in N^1$, and for every $j \in [vi]$, $\zeta_{i,j} \in T_\Delta(Y)$, then for every $q_1,..., q_n \in Q$ and every $p_1,..., p_m \in Q$, the rule $[A, (q_1,..., q_n) \to q](y_1,..., y_n) \to$ *if* root $= \sigma$ *then* $\zeta'$ is in $R$, where $\zeta' = t'[z_i \leftarrow [B_i, (p_{i,1},..., p_{i,vi}) \to p_i]\langle \text{sel}_{\kappa i}\rangle(\zeta'_{i,1},..., \zeta'_{i,vi}); i \in [m]]$ and for every $i \in [m]$ and $j \in [vi]$,

$$\zeta_{i,j}[y_k \leftarrow q_k(y_k); k \in [n]] \Rightarrow^*_{M_2(Y_n)} p_{i,j}(\zeta'_{i,j})$$

and

$$t[z_i \leftarrow p_i(z_i); i \in [m]][y_k \leftarrow q_k(y_k); k \in [n]] \Rightarrow^*_{M_2(Z_m \cup Y_n)} q(t').$$

For a finite set $Z$, $M_2(Z)$ denotes the relabeling $(Q, \Delta, \Lambda \cup Z, R_2)$. If $A = A_{\text{in}}^1$, then additionally for every $q \in Q$, if $[A, (\,) \to q] \to$ *if* root $= \sigma$ *then* $\zeta'$ is in $R$, then the rule $A_{\text{in}} \to$ *if root* $= \sigma$ *then* $\zeta'$ is in $R$. This ends the construction of $M$.

It is obvious that, if $M_1$ is a top-down tree transducer, then $M$ is also a top-down tree transducer. Clearly, for every $A \in N^1$ of rank $n \geqslant 0$, $q, q_1,..., q_n \in Q$, $s \in T_\Sigma$, and $\xi \in T_\Delta(\bar{Y}_n)$, where $\bar{Y}_n = \{\bar{y}_1,..., \bar{y}_n\}$ is a set with elements of rank 0, $[A, (q_1,..., q_n) \to q]\langle s\rangle(\bar{y}_1,..., \bar{y}_n) \Rightarrow^*_{M(\bar{Y}_n)} \xi$ iff there is a $\xi' \in T_\Delta(\bar{Y}_n)$ such that $A\langle s\rangle(\bar{y}_1,..., \bar{y}_n) \Rightarrow^*_{M_1(\bar{Y}_n)} \xi'$ and $\xi'[\bar{y}_k \leftarrow q_k(\bar{y}); k \in [n]] \Rightarrow^*_{M_2(\bar{Y}_n))} q(\xi)$. From this statement $\tau(M_1) \circ \tau(M_2) = \tau(M)$ follows. ∎

Now we can state the generalization of Lemma 5.6 to compositions of transducers.

5.8. THEOREM. *For every* $n \geqslant 1$, $\text{BT(TR)}^n \circ \pi \subseteq \text{RELAB} \circ \pi \circ \text{LIN(one-way)}^n$ *and* $\text{RT(TR)}^n \circ \pi \subseteq \text{RELAB} \circ \pi \circ \text{REG(one-way)}^n$.

*Proof.* The proof of the first inclusion is by induction on $n$, where the case $n = 1$ is proved in Lemma 5.6. Now assume that $\text{BT(TR)}^n \circ \pi \subseteq \text{RELAB} \circ \pi \circ \text{LIN(one-way)}^n$. Then,

$$\begin{aligned}
\text{BT(TR)}^{n+1} \circ \pi &= \text{BT(TR)}^n \circ \text{BT(TR)} \circ \pi \\
&\subseteq \text{BT(TR)}^n \circ \text{RELAB} \circ \pi \circ \text{LIN(one-way)} \qquad \text{by Lemma 5.6} \\
&\subseteq \text{BT(TR)}^n \circ \pi \circ \text{LIN(one-way)} \qquad \text{by Lemma 5.7} \\
&\subseteq \text{RELAB} \circ \pi \circ \text{LIN(one-way)}^n \circ \text{LIN(one-way)} \\
&\quad \text{by induction hypothesis} \\
&\subseteq \text{RELAB} \circ \pi \circ \text{LIN(one-way)}^{n+1}.
\end{aligned}$$

The proof of the second inclusion of this theorem is analogous. ∎

5.2. *Composition Hierarchy*

Before proving the strictness of the composition hierarchy of basic tree transducers, the connection between compositions of basic tree transducers and iterated linear control is shown. In particular, the path languages of images of RECOG under basic tree transducers are proved to be linear.

5.9. THEOREM. (i) *For every* $n \geq 0$, $\pi(\mathrm{BT(TR)}^n(\mathrm{RECOG})) \subseteq \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$

(ii) $\pi(\mathrm{BT(TR)(RECOG)}) \subseteq \mathscr{L}_{\mathrm{LIN}}$.

*Proof.* It follows from Facts 2.2 and 2.1 that $\pi(\mathrm{RELAB(RECOG)}) \subseteq \mathscr{L}_{\mathrm{REG}}$. Then, by Theorem 5.8, $\pi(\mathrm{BT(TR)}^n(\mathrm{RECOG})) \subseteq \mathrm{LIN(one\text{-}way)}^n(\mathscr{L}_{\mathrm{REG}})$, and (i) follows from Lemma 2.5. Part (ii) follows from (i) (for $n=1$) and from $\mathrm{LIN(one\text{-}way)}(\mathscr{L}_{\mathrm{REG}}) = \mathscr{L}_{\mathrm{LIN}}$ (Fact 2.6). ∎

This theorem shows the connection between basic tree transducers and controlled linear grammars as expected in [Vog1]. We note that the classes, that are related in Theorem 5.9 by inclusions, are actually very close to each other. If we would assume that the languages generated by $\pi(\mathscr{L})$-controlled linear grammars are path languages of monadic tree languages, then we could easily prove the inclusion in the other direction. Note that the class $\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$ is the class of $n$-iterated one-turn pushdown languages, i.e., $\mathscr{L}(\mathrm{P}_{1t}^n)$ [Vog1] (recall that $\mathscr{L}(S)$ denotes the class of languages accepted by usual $S$-automata).

Now we turn to the composition hierarchy of basic tree transducers. The strictness of this hierarchy is based on the inclusion $\pi(\mathrm{BT(TR)}^n(\mathrm{RECOG})) \subseteq \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$ and on the known properness of the hierarchy $\{\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}}) \mid n \geq 0\}$. For every $n \geq 0$, we define a language $\tilde{L}_n$ and by showing that $\tilde{L}_n \notin \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$ and that $\tilde{L}_n \in \pi(\mathrm{BT(TR)}^{n+1}(\mathrm{RECOG}))$, we can conclude from Theorem 5.9 even the properness of the hierarchy $\{\mathrm{BT(TR)}^n(\mathrm{RECOG}) \mid n \geq 0\}$ of images of RECOG under compositions of basic tree transducers.

The language $\tilde{L}_n$ is very close to the language $L_n$ used by Khabbaz to prove the properness of the hierarchy $\{\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{CF}}) \mid n \geq 0\}$ [Kha1]. Recall that $L_n = \{a^k c_1 a^k c_2 a^k c_3 \cdots a^k c_{\exp(n+1)} \mid k \geq 0\}$, where $\exp(m) = 2^m$, and that $L_n \in \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{CF}}) - \mathrm{CTRL}_{n-1}(\mathrm{LIN}, \mathscr{L}_{\mathrm{CF}})$. Actually, Khabbaz even proved that $L_n \in \mathrm{CTRL}_{n+1}(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$. Note that, in particular, $L_n \notin \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$, because $\mathrm{CTRL}(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}}) \subseteq \mathscr{L}_{\mathrm{CF}}$.

Now define, for every $n \geq 0$, the language $\tilde{L}_n = \{(\#, 1)(a, 1)^k (c_1, 1)(a, 1)^k (c_2, 1) \cdots (a, 1)^k (c_{\exp(n+1)}, 0) \mid k \geq 0\}$.

5.10. LEMMA. *For every* $n \geq 0$, $\tilde{L}_n \notin \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$.

*Proof.* Assume that $\tilde{L}_n \in \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$. Since $\mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$ is a full semi-AFL (cf. Fact 2.4), also $h(\tilde{L}_n) \in \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$, where $h$ is the

homomorphism which maps $(\#, 1)$ to the empty string and replaces every symbol $(x, i)$ by $x$. But $h(\tilde{L}_n) = L_n \notin \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$, which is a contradiction. ∎

Next we realize $\tilde{L}_n$ as the path language of the image of a recognizable tree language under $\tau$ for some $\tau \in \mathrm{D_t BT(TR)}^{n+1}$. The desired basic tree transducers $M_1, \ldots, M_{n+1}$ are just the "monadic tree versions" of the controlled linear grammars, which Khabbaz used in [Kha1], to generate $L_n$.

5.11. THEOREM. *For every* $n \geqslant 0$, $\mathrm{D_t BT(TR)}^{n+1}(\mathrm{RECOG}) - \mathrm{BT(TR)}^n(\mathrm{RECOG}) \neq \varnothing$.

*Proof.* Let $n \geqslant 0$. First we construct a ranked alphabet $\Sigma$, a tree language $L \in \mathrm{RECOG}$, and a $\tau \in \mathrm{D_t BT(TR)}^{n+1}$ such that $\tilde{L}_n = \pi_\Sigma(\tau(L))$. Then define $K = \tau(L)$. By definition, $K \in \mathrm{D_t BT(TR)}^{n+1}(\mathrm{RECOG})$. Assume that $K \in \mathrm{BT(TR)}^n(\mathrm{RECOG})$. Then, by Theorem 5.9(i), $\pi_\Sigma(K) = \pi_\Sigma(\tau(L)) = \tilde{L}_n \in \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$. This contradicts Lemma 5.10, and hence, $K \notin \mathrm{BT(TR)}^n(\mathrm{RECOG})$.

For every $i$ with $0 \leqslant i \leqslant n$, define the ranked alphabet $\Sigma_i = \{\#, a\} \cup \{c_1, c_2, \ldots, c_{\exp(i)-1}, c_{\exp(i)}\}$, where every symbol has rank 1 except $c_{\exp(i)}$, which has rank 0. Define the total deterministic basic tree transducer $M_i = (\{A_{\mathrm{in}}^{(0)}, A^{(1)}\}, \Sigma_i, \Sigma_{i+1}, A_{\mathrm{in}}, R_i)$, where $R_i$ contains the following rules:

$A_{\mathrm{in}} \to \textit{if } \mathrm{root} = \# \textit{ then } \#(A\langle \mathrm{sel}_1 \rangle(c_{\exp(i+1)}))$,

$A(y_1) \to \textit{if } \mathrm{root} = a \textit{ then } a(A\langle \mathrm{sel}_1 \rangle(a(y_1)))$, for every $j \in [\exp(i) - 1]$,

$A(y_1) \to \textit{if } \mathrm{root} = c_j \textit{ then } c_j(A\langle \mathrm{sel}_1 \rangle(c_{\exp(i+1)-j}(y_1)))$, and

$A(y_1) \to \textit{if } \mathrm{root} = c_{\exp(i)} \textit{ then } c_{\exp(i)}(y_1)$.

Moreover, for every $\sigma \in \Sigma_i - \{\#\}$, $R_i$ contains the dummy rules

$A_{\mathrm{in}} \to \textit{if } \mathrm{root} = \sigma \textit{ then } c_{\exp(i+1)}$,

$A(y_1) \to \textit{if } \mathrm{root} = \# \textit{ then } c_{\exp(i+1)}$, and $A(y_1) \to \textit{if } \mathrm{root} = a \textit{ then } c_{\exp(i+1)}$.

The dummy rules guarantee the totality of $M_i$. Obviously, for $\Sigma = \Sigma_{n+1}$, the language $L = \{\#a^k c_1 \mid k \geqslant 0\}$ of monadic trees (here represented as strings), and $\tau = \tau(M_0) \circ \tau(M_1) \circ \cdots \circ \tau(M_n)$, the equality $\tilde{L}_n = \pi_\Sigma(\tau(L))$ holds. ∎

Now the properness of the composition hierarchy of basic tree transducers (for both the nondeterministic and the total deterministic case) is an immediate consequence.

5.12. THEOREM. *For every* $n \geqslant 0$,

(i) $\mathrm{BT(TR)}^n(\mathrm{RECOG}) \subsetneq \mathrm{BT(TR)}^{n+1}(\mathrm{RECOG})$ *and* $\mathrm{D_t BT(TR)}^n(\mathrm{RECOG}) \subsetneq \mathrm{D_t BT(TR)}^{n+1}(\mathrm{RECOG})$,

(ii) $\mathrm{BT(TR)}^n \subsetneq \mathrm{BT(TR)}^{n+1}$ *and* $\mathrm{D_t BT(TR)}^n \subsetneq \mathrm{D_t BT(TR)}^{n+1}$.

*Proof.* Part (ii) follows from (i) and (i) follows from Theorem 5.11. ∎

5.3. *Connections between Top-down Tree Transducers, Basic Tree Transducers, and Macro Tree Transducers*

The aim of this section is to prove (by means of arguments on path languages) that macro tree transducers are strictly more powerful than basic tree transducers (cf. also Corollary 3.9 and Theorem 3.10) and that these are strictly more powerful than top-down tree transducers. There is even a tree language in $D_tCFT(TR)(RECOG)$, which cannot be realized as range of compositions of basic tree transducers; similarly, there is a tree language in $D_tBT(TR)(RECOG)$, which is not in $\bigcup\{RT(TR)^n(RECOG) \mid n \geqslant 1\}$. In the sequel, for every $k \geqslant 0$, let $sq(k) = k^2$.

5.13. LEMMA. *The language* $\{\sigma^{sq(k)}a \mid k \geqslant 0\}$ *of monadic trees is contained in* $D_tCFT(TR)(RECOG)$.

*Proof.* Construct the total deterministic CFT(TR)-transducer $M = (N, \Sigma, \Delta, A_{in}, R)$ by $N = \{A_{in}^{(0)}, A^{(1)}, B^{(1)}\}$, $\Sigma = \Delta = \{\sigma^{(1)}, a^{(0)}\}$, and $R$ contains the rules

$A_{in} \to$ *if* root $= \sigma$ *then* $A\langle sel_1\rangle(B\langle sel_1\rangle(B\langle sel_1\rangle(\sigma(a))))$,

$A_{in} \to$ *if* root $= a$ *then* $a$,

$A(y) \to$ *if* root $= \sigma$ *then* $A\langle sel_1\rangle(B\langle sel_1\rangle(B\langle sel_1\rangle(\sigma(y))))$,

$A(y) \to$ *if* root $= a$ *then* $y$,

$B(y) \to$ *if* root $= \sigma$ *then* $\sigma(B\langle sel_1\rangle(y))$, and

$B(y) \to$ *if* root $= a$ *then* $y$.

It is easy to see that $\tau(M) = \{(\sigma^k a, \sigma^{sq(k)}a) \mid k \geqslant 0\}$. Hence, $\{\sigma^{sq(k)}a \mid k \geqslant 0\} = \tau(M)(T_\Sigma)$ is in $D_tCFT(TR)(RECOG)$. ∎

In [Gre4] it is shown that languages in $\bigcup\{CTRL_n(LIN, \mathscr{L}_{REG}) \mid n \geqslant 0\}$ are accepted by checking stack automata. Let $\mathscr{L}(CS)$ denote the class of languages accepted by (one-way) checking stack automata.

5.14. LEMMA. *For every* $n \geqslant 0$, $\pi(BT(TR)^n(RECOG)) \subseteq \mathscr{L}(CS)$.

*Proof.* This follows from Theorem 5.9(i) and Corollary 3.8.1 of [Gre4]. ∎

5.15. THEOREM. (i) $D_tCFT(TR)(RECOG) - \bigcup\{BT(TR)^n(RECOG) \mid n \geqslant 0\} \neq \emptyset$.

(ii) *For every* $n \geqslant 1$, $BT(TR)^n(RECOG) \subsetneq CFT(TR)^n(RECOG)$ *and* $D_tBT(TR)^n(RECOG) \subsetneq D_tCFT(TR)^n(RECOG)$.

*Proof.* Clearly, (ii) follows from (i) and Fact 3.5. By Lemma 5.13, the language $L = \{\sigma^{sq(k)}a \mid k \geqslant 0\}$ is in $D_tCFT(TR)(RECOG)$. Assume that $L$ is in $BT(TR)^n(RECOG)$ for some $n \geqslant 0$. Then, by Lemma 5.14, $\pi_\Delta(L)$ with

$\Delta = \{\sigma^{(1)}, a^{(0)}\}$ is in $\mathscr{L}$(CS), and since $\mathscr{L}$(CS) is a full AFL (Theorem 1.1 of [Gre1]), also $L' = h(\pi_\Delta(L)) = \{\sigma^{sq(k)} \mid k \geqslant 0\}$ is in $\mathscr{L}$(CS), where $h$ is the homomorphism which replaces $(\sigma, 1)$ by $\sigma$ and $(a, 0)$ by $\lambda$. But it is known that $L'$ is not a checking stack language (cf., e.g., [Gre4]). Hence, $L$ is not in BT(TR)$^n$ (RECOG). ▮

To prove the connection between top-down tree transducers and basic tree transducers, we first show that the path languages of the images of RECOG under compositions of top-down tree transducers are regular languages.

5.16. LEMMA. *For every* $n \geqslant 0$, $\pi(\mathrm{RT(TR)}^n\,(\mathrm{RECOG})) \subseteq \mathscr{L}_{\mathrm{REG}}$.

*Proof.* By Facts 2.1 and 2.2, $\pi(\mathrm{RELAB(RECOG)}) \subseteq \mathscr{L}_{\mathrm{REG}}$. Then, by Theorem 5.8, $\pi(\mathrm{RT(TR)}^n\,(\mathrm{RECOG})) \subseteq \mathrm{REG(one\text{-}way)}^n\,(\mathscr{L}_{\mathrm{REG}})$. Now the result follows from Lemma 2.5. ▮

5.17. THEOREM. (i) $\mathrm{D_tBT(TR)(RECOG)} - \bigcup \{\mathrm{RT(TR)}^n\,(\mathrm{RECOG}) \mid n \geqslant 0\} \neq \varnothing$.

(ii) *For every* $n \geqslant 1$, $\mathrm{RT(TR)}^n\,(\mathrm{RECOG}) \subsetneq \mathrm{BT(TR)}^n\,(\mathrm{RECOG})$ *and* $\mathrm{D_tRT(TR)(RECOG)} = \mathrm{D_tRT(TR)}^n\,(\mathrm{RECOG}) \subsetneq \mathrm{D_tBT(TR)(RECOG)}$.

*Proof.* (ii) follows from (i) and Fact 3.5. Note that total deterministic top-down tree transducers are closed under compositions [Rou2]. Consider the total deterministic basic tree transducer $M$ of Example 3.4 and the language $L = \tau(M)(T_\Sigma)$, where $\Sigma$ and $\Delta$ are the input alphabet and the terminal alphabet of $M$, respectively. By definition, $L$ is in $\mathrm{D_tBT(TR)(RECOG)}$. Assume that $L$ is in $\mathrm{RT(TR)}^n\,(\mathrm{RECOG})$ for some $n \geqslant 0$. Then, by Lemma 5.16, $\pi_\Delta(L) \in \mathscr{L}_{\mathrm{REG}}$. Let $h$ be the homomorphism defined in Example 3.4. Since $\mathscr{L}_{\mathrm{REG}}$ is closed under homomorphisms, $h(\pi(L)) = \{w\$w^{\mathrm{R}} \mid w \in \{\sigma, \delta\}^* \alpha\} \in \mathscr{L}_{\mathrm{REG}}$ which is not true. Hence, $L$ is not in $\mathrm{RT(TR)}^n\,(\mathrm{RECOG})$ for any $n$. ▮

## 6. CONCLUSION

We have investigated the class BT(TR) of translations induced by basic tree transducers. These devices form a natural restriction of macro tree transducers. For two slight extensions of BT(TR), viz., $\mathrm{BT_{reg}(TR)}$ and $\mathrm{BT_{fin}(TR)}$, a characterization in terms of one-turn pushdown machines is proved: $\mathrm{BT_{reg}(TR)} = \mathrm{RT(P_{1t}(TR))}$ and $\mathrm{BT_{fin}(TR)} = \mathrm{RT(P_{1t,bex}(TR))}$. In the total deterministic case, the composition of basic tree transducers is equivalent to iterated one-turn pushdown machines, i.e., $\mathrm{D_tBT(TR)}^n = \mathrm{D_tRT(P}_{1t}^n\mathrm{(TR))}$.

The study of tree-to-path translations of (compositions of) basic tree transducers has provided a formal relationship with (iterated) control on linear grammars, i.e., $\pi(\mathrm{BT(TR)}^n\,(\mathrm{RECOG})) \subseteq \mathrm{CTRL}_n(\mathrm{LIN}, \mathscr{L}_{\mathrm{REG}})$; hence, languages in $\pi(\mathrm{BT(TR)}^n$ (RECOG)) are accepted by one-way $\mathrm{P}_{1t}^n$ automata. The formalism of controlled

linear grammar (viewed as "grammar directed translators" [Gre3]) coincides with the concept of basic tree transducer, in which every ranked alphabet is monadic.

Thus, the machine characterization result and the study of tree-to-path translations of basic tree transducers links the following three formalisms

— basic tree transducers

— one-turn pushdown machines

— controlled linear grammars.

The next diagram shows the connections between compositions of top-down tree transducers, of basic tree transducers, and of macro tree transducers. Solid ascending lines denote strict inclusion.

The correctness of the diagram in Fig. 2 (with respect to strictness of inclusions) follows immediately from

(1) $\mathrm{RT(TR)}^n \subsetneq \mathrm{BT(TR)}^n$ (by Theorem 5.17(ii)),

(2) $\mathrm{BT(TR)}^n \subsetneq \mathrm{CFT(TR)}^n$ (by Theorem 5.15(ii)),

(3) the strictness of the composition hierarchy of top-down tree transducers (by Theorem 3.14 of [Eng4]),

(4) $\mathrm{BT(TR)}^n \subsetneq \mathrm{BT(TR)}^{n+1}$ and $\mathrm{D_tBT(TR)}^n \subsetneq \mathrm{D_tBT(TR)}^{n+1}$ (by Theorem 5.12),

(5) the strictness of the composition hierarchy of nondeterministic and total deterministic macro tree transducers (by Theorem 4.16 of [EngVog1]), and

(6) the fact that, for $X \in \{\mathrm{RT(TR)}^n, \mathrm{BT(TR)}^n, \mathrm{CFT(TR)}^n\}$, $\mathrm{D_t}X$ is a class of mappings, whereas $X$ contains partial functions.

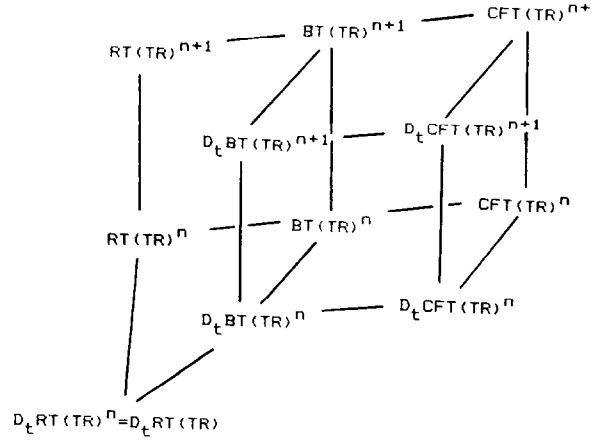


FIG. 2. Inclusion diagram for classes of translations ($n \geq 1$).

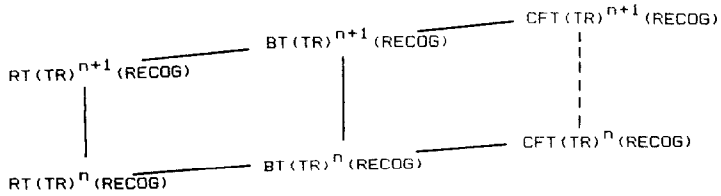


FIG. 3. Inclusion diagram for images of RECOG under classes of translations ($n \geq 1$).

To show the correctness of the diagram of Fig. 2 also with respect to the incomparability of non-connected classes, it remains to prove that for every $n \geq 1$,

(i) $\mathrm{RT(TR)}^{n+1} - \mathrm{CFT(TR)}^n \neq \varnothing$

(ii) $\mathrm{D_t BT(TR)}^{n+1} - \mathrm{CFT(TR)}^n \neq \varnothing$.

The other "incomparability results" follow from Lemma 5.11, Theorem 5.15(i), and Theorem 5.17(i). Note that (i) and (ii) cannot be proved by height arguments and it also seems impossible to prove the results by arguments on path languages.

Figure 3 partially improves Fig. 2 by showing the connections of images of RECOG under compositions of top-down tree transducers, basic tree transducers, and macro tree transducers, where solid ascending lines again denote strict inclusion.

Also this diagram is correct with respect to the strictness of inclusions. This also holds for the total deterministic case. We conjecture that the dotted line also represents strict inclusion, i.e., for every $n \geq 0$, $\mathrm{CFT(TR)}^n\mathrm{(RECOG)} \subsetneq \mathrm{CFT(TR)}^{n+1}\mathrm{(RECOG)}$. (Note that in [Dam] $\mathrm{D_t CFT(TR)}^n\mathrm{(RECOG)} \subsetneq \mathrm{D_t CFT(TR)}^{n+1}\mathrm{(RECOG)}$ is proved.) For this inclusion and the incomparability of unconnected classes, it remains to prove a statement stronger than (i).

(i') $\mathrm{RT(TR)}^{n+1}\mathrm{(RECOG)} - \mathrm{CFT(TR)}^n\mathrm{(RECOG)} \neq \varnothing$.

The statements (i), (i'), and (ii) may be the subject of further research.

## ACKNOWLEDGMENT

The author is grateful to Joost Engelfriet for helpful comments and improving suggestions on earlier versions of this paper.

## REFERENCES

[Aho1] A. V. AHO, Indexed grammars, an extension of context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647–671.

[Aho2] A. V. AHO, Nested stack automata, *J. Assoc. Comput. Mach.* **16** (1969), 383–406.

[Cou] B. COURCELLE, A representation of trees by languages I and II, *Theoret. Comput. Sci.* **6** (1978), 255–279; **7** (1978), 25–55.

[CouFra1] B. COURCELLE AND P. FRANCHI-ZANNETTACCI, Attribute grammars and recursive program schemes I, II, *Theoret. Comput. Sci.* **17** (1982), 163–191, 235–257.

[CouFra2] B. COURCELLE AND P. FRANCHI-ZANNETTACCI, On the equivalence problem of attribute systems, *Inform. Control* **52** (1982), 275–305.

[Dam] W. DAMM, The IO- and OI- hierarchies, *Theoret. Comput. Sci.* **20** (1982), 95–206.

[DamGoe] W. DAMM AND A. GOERDT, An automata-theoretic characterization of the OI-hierarchy, *Inform. Control* **71** (1986), 1–32.

[DamGue] W. DAMM AND I. GUESSARIAN, Combining $T$ and level $n$, *in* "Proceedings, 9th Mathematical Foundations of Computer Sciences 1981," Lecture Notes in Computer Sci. Vol. 118, pp. 262–270, Springer-Verlag, New York/Berlin, 1981.

[Dow] P. J. DOWNEY, "Formal Languages and Recursion Schemes," Ph.D. thesis, Rep. TR-16-74, Harvard University, Cambridge, Mass., 1974.

[DusPar] J. DUSKE AND R. PARCHMANN, Linear indexed languages, *Theoret. Comput. Sci.* **32** (1984), 47–60.

[Eng1] J. ENGELFRIET, Bottom-up and top-down tree transformations—a comparison, *Math. Systems Theory* **9** (1975), 198–231.

[Eng2] J. ENGELFRIET, Top-down tree transducers with regular look-ahead, *Math. Systems Theory* **10** (1977), 289–303.

[Eng3] J. ENGELFRIET, Some open questions and recent results on tree transducers and tree languages, *in* "Formal Language Theory; Perspectives and Open Problems" (R. V. Book, Ed.), Academic Press, New York, 1980.

[Eng4] J. ENGELFRIET, Three hierarchies of transducers, *Math. Systems Theory* **15** (1982), 95–125.

[Eng5] J. ENGELFRIET, Iterated pushdown automata and complexity classes, *in* "Proceedings, 15th STOC, Boston, April 1983," pp. 365–373.

[Eng6] J. ENGELFRIET, "Context-free Grammars with Storage," technical report, 1986, Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands.

[EngSch] J. ENGELFRIET AND E. M. SCHMIDT, IO and OI, *J. Comput. System Sci.* **15** (1977), 328–353; **16** (1978), 67–99.

[EngSchvLe] J. ENGELFRIET, E. M. SCHMIDT, AND J. VAN LEEUWEN, Stack machines and classes of nonnested macro grammars, *J. Assoc. Comput. Mach.* **27** (1980), 96–117.

[EngSlu1] J. ENGELFRIET AND G. SLUTZKI, Bounded nesting in macro grammars, *Inform. Control* **42** (1979), 157–193.

[EngSlu2] J. ENGELFRIET AND G. SLUTZKI, Extended macro grammars and stack controlled machines, *J. Comput. System Sci.* **29** (1984), 366–408.

[EngRozSlu] J. ENGELFRIET, G. ROZENBERG, AND G. SLUTZKI, Tree transducers, L-systems, and two-way machines, *J. Comput. System Sci.* **20** (1980), 150–202.

[EngVog1] J. ENGELFRIET AND H. VOGLER, Macro tree transducers, *J. Comput. System Sci.* **31** (1985), 71–146.

[EngVog2] ([EV]) J. ENGELFRIET, H. VOGLER, Pushdown machines for the macro tree transducer; *Theoret. Comput. Sci* **42** (1986), 251–368.

[EngVog3] J. ENGELFRIET AND H. VOGLER, "High Level Tree Transducers and Iterated Pushdown Machines," Report 85-12, Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, May 1985; Characterization of high level tree transducers, *in* "Proceedings, ICALP, Nafplion, Greece, 1985."

[EngVog4] J. ENGELFRIET AND H. VOGLER, "Look-ahead on Pushdowns," Report 85-14, Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, July 1985, to appear in *Inform. Control.*

[Fil] G. FILE, "The Characterization of Some Language Families by Classes of Indexed Grammars," M.Sc. thesis, Dept. of Comput. Sci., Pennsylvania State U., University Park, Pa., 1977.

[Fis] M. J. FISCHER, "Grammars with Macro-like Productions," Ph.D. thesis, Harvard University, 1968.

[Gin] S. GINSBURG, "Algebraic and Automata-Theoretic Properties of Formal Languages," North-Holland, Amsterdam, 1975.

[GinSpa1] S. GINSBURG AND E. H. SPANIER, Finite-turn pushdown automata, *SIAM J. Control Optim.* **3** (1966), 429–453.

[GinSpa2] S. GINSBURG AND E. H. SPANIER, Control sets on grammars, *Math. Systems Theory* **2** (1968), 159–177.

[Gre1] S. A. GREIBACH, Checking automata and one-way stack languages, *J. Comput. System Sci.* **3** (1969), 196–217.

[Gre2] S. A. GREIBACH, Full AFLs and nested iterated substitution, *Inform. Control.* **16** (1970), 7–35.

[Gre3] S. A. GREIBACH, Control sets on context-free grammar forms, *J. Comput. System Sci.* **15** (1977), 35–98.

[Gre4] S. A. GREIBACH, One way finite visit automata, *Theoret. Comput. Sci.* **6** (1978), 175–221.

[Gue] I. GUESSARIAN, Pushdown tree automata, *Math. Systems Theory* **16** (1983), 237–263.

[Har] M. A. HARRISON, "Introduction to Formal Language Theory," Addison–Wesley, Reading, Mass., 1978.

[HopUll] J. E. HOPCROFT AND J. D. ULLMAN, "Formal Languages and Their Relation to Automata," Addison–Wesley, Reading, Mass., 1979.

[Kha1] N. A. KHABBAZ, A geometric hierarchy of languages, *J. Comput. System Sci.* **8** (1974), 142–157.

[Kha2] N. A. KHABBAZ, Control sets on linear grammars, *Inform. Control.* **25** (1974), 206–221.

[Mas] A. N. MASLOV, Multi-level stack automata, *Problems Inform. Transmission* **12** (1976), 38–43.

[Rou1] W. C. ROUNDS, Tree-oriented proofs of some theorems on context-free and indexed languages, *in* "2nd Symposium on Theory of Computing, 1970," pp. 109–116.

[Rou2] W. C. ROUNDS, Mappings and grammars on trees, *Math. Systems Theory* **4** (1970), 257–287.

[Roz] G. ROZENBERG, Extension of tabled 0L-systems and languages, *Internat. J. Comput. Inform. Sci.* **2** (1973), 311–336.

[RozSal] G. ROZENBERG AND A. SALOMAA, "The Mathematical Theory of L Systems," Academic Press, New York, 1980.

[Sal] A. SALOMAA, "Formal Languages," Academic Press, New York, 1973.

[Tha] J. W. THATCHER, Generalized$^2$ sequential machine maps, *J. Comput. System Sci.* **4** (1970), 339–367.

[ThaWri] J. W. THATCHER AND J. B. WRIGHT, Generalized finite automata theory with an application to a decision-problem of second-order logic, *Math. Systems Theory* **2** (1968), 58–81.

[vLe] J. VAN LEEUWEN, Notes on pre-set pushdown automata, *in* "L Systems" (G. Rozenberg and A. Salomaa, Eds.) Lecture Notes in Computer Science Vol. 15, pp. 177–188, Springer-Verlag, New York/Berlin, 1974.

[Vog1] H. VOGLER, "Iterated Linear Control and Iterated One-turn Pushdowns," Report 85-04, Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, March 1985; *Math. Systems Theory*, in press.

[Vog2] H. VOGLER, "The OI-hierarchy is closed under control," technical report, Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, August 1985; *in* "Proceedings, MFCS 1986, Bratislava, CSSR."